LINMA2472 – Algorithms in Data Science Homework II Part I : Graph Kernel

Brieuc Pinon[1]

October 20, 2022

1 Introduction

In this homework, we will apply visualization and Machine Learning classification tools on data represented by graphs. We will define kernels on the graphs to allow their manipulation by algorithms for which the kernel trick can be used. Moreover, by defining a kernel we implicitly define a Euclidean distance between the graphs. This allows methods purely based on Euclidean distances.

The data we will use comes from chemistry and bio-chemistry.

We will focus on one graph kernel in this homework, the Weisfeiler-Lehman subtree kernel (Shervashidze et al., 2011). But note that many kernels have been defined on graphs in the literature (Kriege et al., 2020), this is just one that is classical, efficient to compute, and that gives good generalization results empirically.

We explain the data and the Weisfeiler-Lehman subtree kernel, then define your tasks. Note that you will rely on a library to compute the WL subtree kernel, but we ask you to understand it and ask brief analysis questions.

2 Graphs from chemistry/bio-chemistry

2.1 Datasets

We will work with 3 datasets[2]:

- The *MUTAG* dataset consists of 188 chemical compounds divided into two classes according to their mutagenic effect on a bacterium.

  The chemical data was obtained from `http://cdb.ics.uci.edu` and converted to graphs, where vertices represent atoms and edges represent chemical bonds. Explicit hydrogen atoms have been removed and vertices are labeled by atom type and edges by bond type (single, double, triple or aromatic). Chemical data was processed using the Chemistry Development Kit (v1.4).

- *ENZYMES* is a dataset of protein tertiary structures obtained from Borgwardt et al. (2005) consisting of 600 enzymes from the *BRENDA* enzyme database (Schomburg et al., 2004).

  In this case the task is to correctly assign each enzyme to one of the 6 EC top-level classes.

- *NCI1* represents a balanced subset of datasets of chemical compounds screened for activity against non-small cell lung cancer and ovarian cancer cell lines respectively (Wale et al. (2008) and `http://pubchem.ncbi.nlm.nih.gov`).

2.2 From proteins to graphs

How do we pass from a chemistry or bio-chemistry prediction problem on molecules to a graph classification problem? We give the answer for the *ENZYMES* dataset. A rough sketch of the transformation for proteins is represented on Figure 1. Secondary structure elements (SSEs) are represented by nodes. SSEs correspond to local 3D structures such as helices, sheets and turns. Then edges are added, those can either correspond to close neighbors in the 3D folding (3 closest SSEs in distance for example) or to proximity in the original amino-acid chain.

Then several types of information can be integrated into the graph through labels on the nodes and edges. For example, the nodes can take label to represent the type of SSE (helices, sheets and turn) and their chemical

[1] brieuc.pinon@uclouvain.be

[2] credits for the descriptions to Kersting et al. (2016)

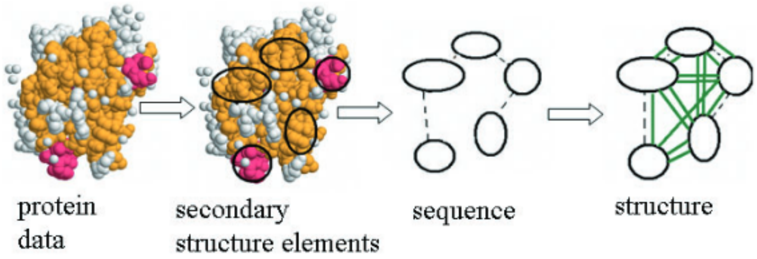

Figure 1: Passing from a protein to a graph. Nodes for SSEs. Dashed lines for proximity in the amino-acid chain, and solid lines for structural proximity. Credit to Borgwardt et al. (2005).

properties (e.g. hydrophobicity). Edge labels can indicate its origin (structural or from the original chain) and distances (in amino-acids or in Angström).

In this work, we will focus on integer labels on nodes, but the algorithms can be extended to take into account more complex labels and also labels on edges.

3 Weisfeiler-Lehman subtree kernel

The Weisfeiler-Lehman subtree kernel that we will present and use is one instance of the Weisfeiler-Lehman family of kernels. To define this subtree version, we need a first (very simple) kernel on labeled graphs.

3.1 A simple kernel

Let be two labeled graphs G_1 and G_2 with node sets V_1 and V_2, integers labels ℓ_1^i and ℓ_2^j respectively where $i \in V_1$ and $j \in V_2$. We define the vertex histogram kernel as

$$k_{\mathrm{VH}}(G_1, G_2) = \sum_{i \in V_1} \sum_{j \in V_2} \delta(\ell_1^i, \ell_2^j) \tag{1}$$

with $\delta(w, z) = 1$ if w and z are equal and 0 otherwise.

This kernel is thus a simple statistic on the similarity of the distribution of node labels on the two graphs. Remark that it can be rewritten as a scalar product. This kernel completely ignore the edges and thus the structure of the graph.

3.2 The family of Weisfeiler-Lehman kernels

To account for the graph structure, we will borrow an idea coming from graph isomorphism checking. The idea is to put in a node label information about its neighbors. If we repeat this enhancing of the label iteratively we will progressively integrate information about the direct neighboring nodes, then the neighbors' neighbors, the neighbors' neighbors' neighbors, etc. thus increasing in the label information about the graph structure.

Here is the formal definition of one iteration starting with some graph G^h with node set V and label ℓ^i for $i \in V$. We will produce a new graph G^{h+1} from this iteration with the same node set and edges, but with new labels.

1. For each node construct $M^i = \{\ell^j | j \text{ neighbour of } i \text{ in } G^h\}$, this is a set of the neighboring labels.
2. Construct the tuple (ℓ^i, M^i) for each node. This adds the information already accumulated at the node itself.
3. Convert the tuples, (ℓ^i, M^i), into integers with an injective function f. This is just to preserve integers that are easy to manipulate, rather than tuples with sets.
4. Set the new labels $\ell^i = f((\ell^i, M^i))$ for the new graph G^{h+1}.

See Figure 2 for an example.

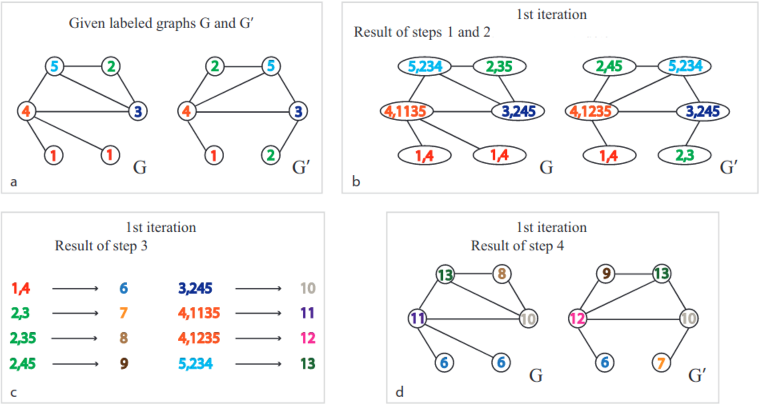

Figure 2: First iteration of the WL algorithm on two labeled graphs. Credit Shervashidze et al. (2011).

3.3 The full algorithm

The WL algorithm produces a new labeled graph at each iteration, and thus a sequence of graphs $G^0, G^1, \ldots, G^H$ for H iterations starting with G^0.

The WL subtree kernel applies the WL algorithm on each graph for H iterations, then compute the sum of the k_{VH} kernel applied iteration by iteration.

$$k_{\mathrm{WLsubtree}}(G_1, G_2) = \sum_{h=0}^{H} k_{\mathrm{VH}}(G_1^h, G_2^h). \tag{2}$$

4 Tasks

A pre-completed Jupyter notebook is given, which loads several libraries of interest and the datasets we will use. You will complete this notebook to address the defined tasks when an implementation is required. Theoretical tasks will be answered in a separate *pdf* report.

Three datasets are used for the tasks *MUTAG*, *ENZYMES* and *NCI1*. When a visualization or a classification is demanded, apply your code on all three of them.

4.1 Computing the kernels

4.1.1 Weisfeiler-Lehman subtree complexity

What is the asymptotic computational complexity of the Weisfeiler-Lehman subtree kernel described above when applied on two graphs of maximum N nodes and E edges for H iterations? Be clear on the data-structures you use.

4.1.2 Compute the kernels

Using the Grakel library, compute the pairwise kernels between all the graphs in the dataset (for an arbitrary number of iteration H).

4.1.3 Explicit features

As explained in the course, kernels, under mild condition, have an implicit embedding space. For the Weisfeiler-Lehman subtree kernel, such an embedding can be made explicit. What is it?

4.1.4 Explicit embedding versus kernel

A justification for the use of kernels is that the implicit embedding space has more dimensions than the number of samples in the datasets, and thus, it is computationally easier to work with the kernels.

Compute the rank of the WL subtree kernel matrix for the three datasets and $H = 10$ iterations.

Then infer and justify a lower-bound on the dimension of the implicit embedding space based on these ranks.

4.2 Visualization

4.2.1 Kernel centralization

PCA is computed on centered (meaning zero-mean) data. The same is true for kernel-PCA. We can center the data in the embedding space by applying the following formula:

$$\tilde{K} = K - \mathbf{1}^N K - K\mathbf{1}^N + \mathbf{1}^N K \mathbf{1}^N, \tag{3}$$

where K is the kernel matrix, $\tilde{K}$ is the centered kernel matrix, N is the number of samples, $\mathbf{1}^N$ is a matrix of size (N, N) with all entries set to $1/N$.

Apply this formula on your kernel matrix to prepare the application of kernel-PCA.

Prove in your report that $\tilde{K}$ is centered. More formally, let $\phi(G)$ be the vector corresponding to the graph G in a feature space consistent with the kernel; let $\bar{\phi}$ be $\frac{1}{N}\sum_{i=1}^{N}\phi(G_i)$. Prove that for all $i, j \in \{1, \ldots N\}$

$$\tilde{K}_{i,j} = \langle \phi(G_i) - \bar{\phi}, \phi(G_j) - \bar{\phi} \rangle . \tag{4}$$

4.2.2 kernel-PCA implementation

Refer to the slides to implement kernel-PCA from scratch given a precomputed centered kernel matrix.

4.2.3 kernel-PCA visualization

Plot the graphs using the 2-dimensional embeddings defined by the two first principal components.

4.2.4 Distance

Compute the pairwise distances between all the graphs in the dataset as $\sqrt{k(G_1, G_1) + k(G_2, G_2) - 2k(G_1, G_2)}$. Prove that this is the classical Euclidean distance in the implicit embedding space of the kernel $k(G_1, G_2)$.

4.2.5 tSNE

Relying on the pairwise distances just computed, apply tSNE using Scikit-learn. Explain briefly tSNE. (You have to do your own research for this method.)

4.2.6 Comparison

Compare the visualizations obtained with kernel-PCA and tSNE.

4.3 Classification

4.3.1 A simple baseline

Estimate the accuracy of the best constant model on the three datasets. A constant model always output the same class without even looking at the input.

4.3.2 Support Vector Machines (SVM)

Split the dataset into a training and a testing partition (80/20).

Find an optimal SVM classifier with the WL subtree kernel on the training dataset. You can use the Scikit-learn library. The hyperparameters, we will consider, are C which inverse proportionally weight a quadratic regularization on the parameters and H the number of iterations in the WL subtree kernel. You can set $C = 1e2$ and $H = 3$ for this first model.

Assess the accuracy of your model on the test partition.

4.3.3 Select hyperparameters

Perform a grid search over the space of hyperparameters $C = \{1e{-}5, 1e{-}4, \dots, 1e4\}$, $H = \{1, 2, \dots, 10\}$ and select the best combination according to its performance on a 10-fold cross validation on the previously isolated training set. You can use Scikit-learn *cross_val_score* function to help you[3].

4.4 Observations

Comment on the classification accuracy and the plots obtained in the visualization part for the three datasets. In particular, do you see a relationship between the two? We do not expect a lengthy description of what you observe, one or two paragraphs are enough.

5 Guidelines

Submit your completed notebook in *.ipynb* and *.pdf* format, with your *.pdf* report in a *zip* file on Moodle for the 13 November at midnight. You should do it by groups of 2/3, registering a group is done on Moodle.

Be as clear and concise as possible in your answers.

Questions on the dedicated Moodle forum or to brieuc.pinon@uclouvain.be.

References

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56.

Kersting, K., Kriege, N. M., Morris, C., Mutzel, P., and Neumann, M. (2016). Benchmark data sets for graph kernels.

Kriege, N. M., Johansson, F. D., and Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, 5(1):1–42.

Schomburg, I., Chang, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., and Schomburg, D. (2004). Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1):D431–D433.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9).

Wale, N., Watson, I. A., and Karypis, G. (2008). Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375.

[3] If you do not know about cross validation this link explains it : https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation