Learning graph similarity measures using the Weisfeiler-Lehman label hierarchy

Fabrice Beaumont

University of Bonn, Germany

Abstract. In this expose I am outlining the research questions and research strategy of my master thesis. In my thesis, I am going to investigate the effect of learning such similarities between Weisfeiler Lehman labels and using these in the definition of a graph similarity measure, structured similarly to the Wasserstein Weisfeiler Lehman graph kernel proposed in 2019 by Togninalli et al. The goal is to discuss, whether learning such similarities can improve the flexibility of the resulting graph similarity measure. The machine learning will be restricted in the sense that the ground distance is set as a a tree metric on the Weisfeiler Lehman label hierarchy tree. The learning procedure is to iteratively updated the edge weights in this tree and thus to change the resulting distance between the Weisfeiler-Lehman labels used.

Keywords: Weisfeiler Lehman · Tree Metric · Graph Classification · Metric Learning · Wasserstein distance. . . .

1 Introduction

A lot of data can be represented as graphs. Common examples are social and sensor networks, program flows and structures in chemo- and bio-informatics (e.g. molecules, proteins, genes and chemical compounds). Based on functional properties of the graphs one may define classifications on them. To extend the classification of known graph instances to unknown instances is one of many different challenges when working with such graph data. Such classification can indicate, for example, whether a particular molecule can trigger mutations or not. In this framework it is usually assumed, that graphs with similar structure have similar properties and therefore should be classified similarly. Therefore, it is important to quantify the similarity between graphs [15].

The most coarse similarity measure is binary, and decides, if two given graphs are isomorphic (topological identical) or not. The computation of this measure is in NP [5]. Furthermore, the task of testing weather for two given graphs one is isomorphic to a subgraph of the other one, is NP-complete [5]. And the seemingly reasonable approach to restrict the isomorphism test to a unique subgraph, like the largest common subgraph, is of limited use, since finding such largest common subgraphs is a NP-complete problem, as well [5].

To define a finer, less binary similarity measure, several approaches have been made. Naturally, they rely on inexact matchings of two graphs. Some examples

are based on the graph edit distance [2, 4], optimal assignment kernels [4], the skew spectrum [10] or the graphlet spectrum [11, 14]. In the context of similarity measures, graph kernels have the desirable property that they allow to apply a larger toolbox of kernelized learning methods on graph structured data. By restricting the kernels to substructures of graphs, they can be computable in polynomial time [7]. This restriction however may at the same time limit their ability to capture complex characteristics of the graphs. That is why most proposed variants do not generalize well to graphs with high-dimensional continuous vertex attributes [15]. In 2019, Togninalli et al. established their Wasserstein Weisfeiler-Lehman (WL) Graph Kernels as the new state of the art. These kernels are capable of processing both categorical labels and continuous attributes by utilizing the the Hamming distance and Euclidean distance as ground metric for the Wasserstein distance [16]. However, these rigid, rather arbitrary definition does not permit to adjust the distances between the labels (or attributes) used in the graph. Such adjusted distances can be useful to introduce-application specific knowledge about the similarity of the original labels (or attributes) or even whole substructures - represented by the Weisfeiler-Lehman labels. Since such application-based differences may not be known or only partially known, one may try to use machine learning to find such database-specific distances. If this possible, the learned distances could potentially reveal unknown application-based knowledge on the substructures of the used graphs.

In my masters thesis I am going to investigate, how well a measure for similarity can be learned, based on a similarity definition using the WL-labeling hierarchy. The structure of the similarity measure is a Wasserstein distance between the WL-label distribution (WL-features) of two given graphs. The ground distance of the Wasserstein distance is defined as a tree metric on the WL-labels. Thus it highly depends on the edge weights in the hierarchy tree. The research question is, if it is possible to learn how to adjust these edge weights, in order to yield a better similarity measure, compared to a static ground distance definition. The usage of the Wasserstein distance is motivated by Togninalli et al., who successfully used it to construct a similarity measure for graphs with both categorical vertex labels and continuous vertex attributes [16]. As they mention, optimal transport had been used successfully to tackle the graph alignment problem before [18].

Before giving details of the used method in section **??** we take an overview about related work in section 2.

2 Related work

Graph kernels can be categorized into four classes: Graph kernels based on walks [6, 9] and paths [1], limited-size subgraphs like graphlets [8, 14], subtree patterns [13, 12] and combinations of them. One important example for such a combination is the General WL kernel proposed by Nino Shervashidze et. al [15]. This kernel is based on the WL labels and can be combined with any other graph kernel (base kernel), since it simply sums up the base kernels results for every

WL label iteration. The authors themselves present a subtree, edge and shortest path variation in their paper "Weisfeiler-Lehman Graph Kernels" [15].

3 Framework and definitions

Before describing the details of the proposed method, let me introduce some basic definitions. A graph is a tuple $G = (V, E)$ where V is a finite set of vertices and $E \subseteq 2^V$ is a finite set of undirected edges on them. G is called labeled, if there exists a vertex labeling function $\ell : V \to \Sigma_V$ for a finite label alphabet Σ. The labels can be include in the definition of the graph by writing $G = (V, E, \ell)$. G is called weighted if there exists an edge-weight function $w : E \to \mathbb{R}$. Two graphs G_1 and G_2 are isomorphic ($G_1 \equiv G_2$), if there exists a bijective function between their vertices, that preserves all edges and labels. For a graph $G = (V, E)$ and a vertex $v \in V$ we call the set $\mathcal{N}^1(v) := \{u \in V |\ (u, v) \in E\}$ the first neighborhood of v in G. The degree $\delta(v)$ of a vertex $v \in V$ is given as the size of its first neighborhood, that is $\delta(v) := |\mathcal{N}(v)|$. Similarly we define the d-th neighborhood (sometimes referred to as d-hop neighborhood) of v in G (for $1 < d \in \mathbb{N}$) as the set of all vertices that are not v itself, that can be connected with v by a path of length d:

$$\mathcal{N}^d(v) := \left\{u \in V |\ \exists w \in \mathcal{N}^{d-1}(v) \text{ s.t. } (u, w) \in E\right\} \backslash \{v\}$$

Call $\mathcal{N}^d_+(v) := \mathcal{N}^d(v) \cup \{v\}$ the extended d-th neighborhood of v in G.

3.1 Weisfeiler-Lehman labeling scheme

The WL labeling scheme (also called 1-WL vertex embedding, WL color refinement or 2-WL as higher-dimensional WL labeling scheme) is a vertex labeling scheme that maps vertex labels to new WL-labels. These WL-labels encode information about the labels of the neighborhood [17]. The general WL color refinement is described more formally in algorithm 1.

Algorithm 1 WL color refinement

Input: a graph $G = (V, E, \ell)$,
a perfect hash function hash.
Output: a vector representation of G of size $|V|$.

1: $c^0_v \leftarrow \text{hash}(\ell)$ for all $v \in V$
2: **while** $\left(c^k(v)\right)_{v \in V} \neq \left(c^{k-1}(v)\right)_{v \in V}$ **do**
3: $\quad c^k(v) \leftarrow \text{hash}\left(c^{\ell-1}(v), \{\!\{c^{k-1}(w) |\ w \in \mathcal{N}_G(v)\}\!\}\right)$ for all $v \in V$
4: **return** $\{\!\{c^k(v) |\ v \in V\}\!\}$

Note that there are several possible implementations of the hash function and the concatenation step in line 3 [18].

this is either very meta or a strange reference.

In algorithm 1 the color refinement is repeated until the representations do not change, that is until a stable configuration is found. This can be used as an imperfect graph isomorphism test. However, for our purposes, an intermediate compression level of the graph structure may be sufficient. Thus we add as input the desired number of labeling iterations k and execute the commands in the while-loop $k-1$ times. The new set of WL-labels in the i-th iteration (V, c^i) are called the i-th WL-feature of the graph.

Note that after the first iteration, the WL labels encode the information of the labels of the first neighborhood (those reachable with paths of length one). In the d-th iteration, the WL labels encode the information of the labels of the d-th neighborhood, since they are constructed on previously defined WL labels, which in turn encode information about neighbors that are further away. Depending on the graph structure and the vertex degree, the encoded information increases at least linearly but often exponentially in the number of iterations.

Remarks on the implementation When applied to labeled graphs (with a finite set of labels), the labeling scheme can be implemented using integers as WL-labels and with simple perfect hash functions. The method can be implemented as an efficient matrix vector multiplication between the adjacency matrix and the WL-label vector using logarithms ~~and~~ of prime numbers as WL-labels.

3.2 Weisfeiler-Lehman labeling tree (WLLT)

Given k iterations of WL-label refinements on a database of graphs, one can construct a WL-labeling tree (WLLT) of the dependencies or inheritance of the WL labels for this database. Labeled and un-labeled graphs can be treated in the same way, by considering vertex labels as the zeroth WL-labels. With this in mind, use an artificial label to initialize the root of the WLLT. The i-th layer of WLLT vertices (children of the root) consists of the arising WL-labels in the i-th iteration of the label refinement. The complete construction of the WLLT is sketched in algorithm 2.

Algorithm 2 WLLT construction

Input: a graph dataset $\mathcal{G}$,
WL labeling depth k.
Output: WLLT T of height k (unweighted).

```
Initialize the WLLT as a single root T = (r, ∅)
for G in 𝒢 do
    for i = 0 to k do
        Generate the i-th WL-feature F of G
        for l ∈ F do
            if l ∉ T then
                V(T) = V(T) ∪ {l}
                E(T) = E(T) ∪ (p, l) where c^{i-1}(v) = p and c^i(v) = l
```

Remarks on the implementation Note that it is efficient to construct the WLLT while defining the WL-labels. For every iteration of WL labeling, a new layer is added to the WLLT.

Adding edge weights There are several possibilities to compare two given WL-labels, that arise from the same previous WL-label and to define a distance between them. In order to extend such a comparison to any two WL-labels (of the same WL labeling iteration) one use a French Railway Metric (FRM) to define edge weights in the WLLT. These can be used as a tree metric, by adding the weights along any path between two vertices.

More precisely, let T be a WLLT. Let $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ ($n \in \mathbb{N}$) be some distance function between WL-labels and a fixed barycenter (Paris) $P \in \mathbb{R}^n$, which is defined for every parent vertex p in T. Such a barycenter can for example be defined as geometric mean with respect to d:

$$P := \frac{1}{|\mathcal{N}^1(p)|} \sum_{c \in \mathcal{N}^1(p)} x$$

For two children x, y (with (p, x), $(p, y) \in E(T)$) define their tree distance $d_T : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ as:

$$\forall x, y \in \mathbb{R}^n : \qquad d_T(x, y) := \begin{cases} 0 & \text{if } x = y \\ d(x, P) + d(P, y) & \text{otherwise} \end{cases} \tag{1}$$

Now set the edge weights to $w\big((p, x)\big) = w\big((p, y)\big) = 1/2 d_P(x, y)$. For d one can for example use the Hamming distance or the Jaccard distance. Now the distance between any two vertices in the tree is given as the sum of edge weights along the unique path between them. Note, that it is not essential to compute a perfectly pairwise distant barycenter - which is generally a difficult task [3]. Furthermore, the tree metric can be defined in many different ways. For example by altering the definition of the distance on the labels d or the definition of the Paris P. However, the proposed method of learning changes the edge weights anyway. Thus I hope, the resulting similarity measure will be largely independent from the initial edge weight definitions.

Finally the tree metric arises naturally from the definition of the edge weights. The distance between two vertices in the tree is defined as the sum of all edge weights in the path that connects the two vertices.

3.3 Optimal transport - Wasserstein distance

In the proposed method, the Wasserstein distance will be used as graph similarity measure on vector representations of the WL-features of graphs. That is why, I can focus on the discrete, one-dimensional version of the Wasserstein distance.

For two finite sets of vectors $X \in \mathbb{R}^n$ and $X' \in \mathbb{R}^{n'}$ the (discrete) L^1-Wasserstein distance (also known as earth mover distance) is defined as:

$$W_M(X, X') := \min_{P \in \Gamma(X, X')} \langle P, M \rangle \tag{2}$$

M is a cost or distance matrix containing $d(x, x')$ for some ground metric d. $P \in \Gamma$ is often called transport matrix or joint probability. P contains fractions that indicate how to map (transport) the values from X to X' with minimal total cost (transport effort), with respect to M.

As long as the vector representations of the graphs are one-dimensional labels it is $m = 1$. Notice that in the context of the proposed research, M will contain the distances between different WL-labels, given by the tree-metric d.

Ground metric The actual values of the Wasserstein distance depends on the definition of the ground metric d which is used to define the cost matrix M. Togninalli et al. used the normalized Hamming distance for categorical vertex features and the Euclidean distance for continuous vertex features [16]. A key difference between their approach and the one I am going to investigate in this thesis, is to iteratively adapt the used ground distance. More precisely, an initial ground distance will be defined using an initial tree metric on the WL-labeling hierarchy tree. Next, graphs with known classification will be used to train the model in order to adapt the tree metric as needed. This process will be denoted as edge weight learning. It is expected, that this will help to determine the impact of the ground metric on the approach proposed by Togninalli et al. and if it varies depending on the dataset.

3.4 Supervised learning

The evaluation of the proposed method is tied to the supervision of the learning process. The supervision is given by a feedback loop, which evaluates the performance of the current similarity measure, defined by the current set of edge weights in the WLLT. The goal of the learning is to increase the performance of the similarity measure. From this, several degrees of success can be derived. For example:

1. Is it possible to reliably improve the performance of the similarity measure?
2. Is it possible to learn a similarity measure, which performs better than the one used in the Wasserstein Weisfeiler-Lehman Graph Kernels?
3. Is it possible to learn a similarity measure, which performs better than some other kernels?

This kind of evaluation will be used to guide the research since it aligns with the research questions. The evaluation of the similarity measure itself will at least be done analogously to the Wasserstein Weisfeiler-Lehman Graph Kernel by Togninalli et al. That is to use the similarity measure to construct a Laplacian kernel and evaluate its performance using an support vector machine [16]. Besides this, other evaluations of the similarity measures may be considered.

One approach of evaluating the similarity measure is to compare its clustering capabilities to some kind of classification clas(G) for any given graph G. The learning of the ground distance can be derived as sketched in algorithm 3. However, the precise update mechanism may change and several approaches may be

investigated. For example by comparing the quality of the clusters derived from the learned distance to other clustering methods.

Algorithm 3 Ground distance learning

Input: a graph dataset $\mathcal{G}$,
learning rate λ,
number of learning epochs k.
Output: Distance matrix D on $\mathcal{G}$.

1: **for** $i = 0$ to k **do**
2: **for** G_1 and G_2 in $\mathcal{G}$ **do**
3: $D = \mathrm{FRM}_T(G_1, G_2)$ ▷ Ground distance computation
4: $i, j = \mathrm{argmax}\left(\min_{P \in \Gamma}(P, D)\right)$ ▷ Wasserstein distance computation
5: $w\big((i,j)\big) = w\big((i,j)\big) + (-1)^{|c(G_1) - c(G_2)|}\lambda$ ▷ Edge weight update
6: **return** $D = \min_{P \in \Gamma}\left(P,\ \mathrm{FRM}_T(G_1, G_2)\right)$ ▷ Distance matrix on $\mathcal{G}$

4 Research goal

Variations of the described method shall be implemented with the goal to answer the following research questions:

1. Is it possible to reliably increase the predictive power (accuracy) of the similarity measure?
2. Is it possible to learn a similarity measure, which performs better than the one used in the Wasserstein Weisfeiler-Lehman Graph Kernels?

If these questions can be answered positively, several further questions arise naturally and shall be investigated as well:

1. Is it possible to learn a similarity measure, which performs better than some other kernels? Which?
2. Are there example of graph dataset, where the method does not yield in a significant improvement? Can this be explained with application-based knowledge?
3. How adaptable is the usage of the tree-metric as ground distance? Is it possible to learn other distance functions?
4. How does the learning approach perform compared to
   - constant edge weights,
   - different static edge weight initialization (using the Hamming distance, Jaccard distance or others),
   - k-means classification.

4.1 Roadmap

Since the goal is to define and implement a learning method, a training set is needed. By definition, it is possible to construct the training set once and use it for many different configurations of the learning method. It can be expected, that the performance of the similarity measure changes over time with the edge weight updates. This learning or training process shall be monitored in order to elaborate on the success and efficiency of the proposed method. At last, a few implementation choices are not defined jet and it may be beneficial to change them. To what extent the method can be changed successfully shall be investigated as well. These three sub tasks may change, as long as answers to the overall research questions are generated.

it might be useful to follow standard deep nn training procedures

Training set Since the learning of the similarity measure will be limited to changing edge weights in the WLLT, the training set consists of weighted WLLTs and the WL-features of the corresponding dataset. Both can be computed independently and in advance of the actual training. The construction of training sets includes constructing (possibly several) WLLTs of classified graph datasets and WL-features. For computational efficiency, they shall be stored in a file format, including the map of WL-features to the original labels and the WL-features of all graphs.

TODO: A few words on the datasets.

Training The training consists of a supervised feedback loop. In each epoch, the distance between pairs of graphs (of a subset) from the graph dataset is computed. If the graphs have a different classification (supervised knowledge), the distance according to the Wasserstein distance shall be increased. This is done by identifying the most significant WL-label distance with respect to the tree-metric in the computation of the Wasserstein distance. The edge weights along the corresponding path in the WLLT are increased by a finite amount λ. If the graphs have the same classification, the weights are decreased.

how?

Although not specified at this point, it may be desirable to use other kinds of feedback as well. This also will depend on the given datasets and knowledge on them.

Furthermore, the specifics of the target weights of the update and the definition of the update (λ) itself may vary.

Variations Different kinds of feedback loops may require different implementation choices with respect to the definition of the tree-metric, Wasserstein distance, or the update choices. While different choices may be investigated in order to answer the research question properly, a comparison between these choices is desired as well. If possible, such variations shall be set in relation to the different datasets and even the different resulting metrics on the WL-labels.

(*) It would be good to think in more detail about a first variant of the update procedure. You could use the optimal assignment given by the Wasserstein computation and ....

References

[1] Karsten Borgwardt and Hans-Peter Kriegel. "Shortest-Path Kernels on Graphs". In: *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 2005. DOI: 10.1109/icdm.2005.132.

[2] H. Bunke and G. Allermann. "Inexact graph matching for structural pattern recognition". In: *Pattern Recognition Letters* 1.4 (Mar. 1983), pp. 245–253. DOI: 10.1016/0167-8655(83)90033-8.

[3] Samuel Cohen, Michael Arbel, and Marc Peter Deisenroth. "Estimating Barycenters of Measures in High Dimensions". In: *CoRR* abs/2007.07105 (2020).

[4] Holger Fröhlich et al. "Optimal assignment kernels for attributed molecular graphs". In: *Proceedings of the 22nd international conference on Machine learning - ICML '05*. ACM Press, 2005. DOI: 10.1145/1102351.1102380.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., Jan. 1979. 340 pp. ISBN: 0716710455.

[6] Thomas Gärtner, Peter Flach, and Stefan Wrobel. "On Graph Kernels: Hardness Results and Efficient Alternatives". In: *Learning Theory and Kernel Machines*. Springer Berlin Heidelberg, 2003, pp. 129–143. DOI: 10.1007/978-3-540-45167-9_11.

[7] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. "Kernel methods in machine learning". In: *The Annals of Statistics* 36.3 (June 2008). DOI: 10.1214/009053607000000677.

[8] Tamás Horváth, Thomas G?rtner, and Stefan Wrobel. "Cyclic pattern kernels for predictive graph mining". In: *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*. ACM Press, 2004. DOI: 10.1145/1014052.1014072.

[9] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. "Marginalized Kernels Between Labeled Graphs". In: *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*. Ed. by Tom Fawcett and Nina Mishra. AAAI Press, 2003, pp. 321–328.

[10] Risi Kondor and Karsten M. Borgwardt. "The skew spectrum of graphs". In: *Proceedings of the 25th international conference on Machine learning - ICML '08*. ACM Press, 2008. DOI: 10.1145/1390156.1390219.

[11] Risi Kondor, Nino Shervashidze, and Karsten M. Borgwardt. "The graphlet spectrum". In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. ACM Press, 2009. DOI: 10.1145/1553374.1553443.

[12] Pierre Mahé and Jean-Philippe Vert. "Graph kernels based on tree patterns for molecules". In: *Machine Learning* 75.1 (Oct. 2008), pp. 3–35. DOI: 10.1007/s10994-008-5086-2.

[13] Jan Ramon and Thomas Gartner. "Expressivity versus Efficienvy of Graph Kernels". In: *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*. 2003.

[14] Nino Shervashidze and Karsten M. Borgwardt. "Fast subtree kernels on graphs". In: *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*. Ed. by Yoshua Bengio et al. Curran Associates, Inc., 2009, pp. 1660–1668.

[15] Nino Shervashidze et al. "Weisfeiler-Lehman Graph and Kernels". In: *J. Mach. Learn. Res.* 12 (2011), pp. 2539–2561.

[16] Matteo Togninalli et al. "Wasserstein Weisfeiler-Lehman Graph Kernels". In: *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. 2019.

[17] Weisfeiler and Leman. "THE REDUCTION OF A GRAPH TO CANONICAL FORM AND THEALGEBRA WHICH APPEARS THEREIN". In: *Journal of Applied Mathematics and Physics*. 1968.

[18] Keyulu Xu et al. "HOW POWERFULARE GRAPH NEURAL NETWORKS?" In: *International Conference on Learning Representations*. 2019.