Evaluating Graphons as Efficient Graph Embeddings

Carlo Saccardi
KTH Royal Institute of Technology
saccardi@kth.se

Daniel Richards Ravi Arputharaj
KTH Royal Institute of Technology
drra@kth.se

Simone Bonato
KTH Royal Institute of Technology
bonato@kth.se

Vishal Nedungadi
KTH Royal Institute of Technology
vishaln@kth.se

1 Introduction

With the rise of social media platforms, there is an increased interest in the field of graph data analysis. In simple form, a graph is constructed to explicate the relationship between entities. The entities are represented as nodes and the relationship between them is constructed as edges. It enables us to work with data that do not necessarily live in the euclidean space.

Naturally, this makes performing traditional machine learning approaches on graphs non-trivial. Broadly speaking, the general interest lies in node classification, link prediction and clustering. For our purposes, we are concerned with clustering graph data. To this end, numerous methods have been proposed to build graph representation that preserves the highest degree of information (e.g, relationship between nodes, general structure of the graph). Of these, embedding graph data at various level seems to have gained traction in the industry. Algorithm like Graph2Vec[1] construct graph embeddings of fixed length in an unsupervised manner. While popular, generating embedding through this method takes considerable amount of time. To overcome this shortcoming, we investigate the use graphons as a possible embedding tool. In statistical modeling, Exchangeable graphs can be represented using non-parametric graphons. They are extremely useful as they can capture the structure of any large graph and can be used to generate dense graphs.

2 Related Work

Our work is mainly inspired by two papers [2] and [3] . The mentioned works propose methods for clustering multiple graphs, without vertex correspondence. Specifically, [2] introduces a novel graph distance based on sorting-and-smoothing graphon estimators. Graphons are symmetric bivariate functions that represent the limiting structure for a sequence of graphs with increasing number of nodes, but can be also viewed as a nonparametric statistical model for exchangeable random graphs. The latter perspective is useful for the purpose of machine learning since it allows us to view the multiple graphs as random samples drawn from one or more graphon models. As already mentioned, [2] propose a distance between two networks, that do not have vertex correspondence and could have different number of vertices.

We view the graphs as random samples from (unknown) graphons, and propose a graph distance that estimates the L2-distance between the graphons. This distance is inspired by the sorting-and-smoothing graphon estimator proposed by [3].This latter graphon estimator consists of two steps: In the first step, the empirical degrees are sorted and the nodes of the graph are rearranged for a canonical ordering. In the second step, the histogram of the sorted graph is computed, and the histogram is smoothed by a total variation minimization. The estimator returned by the SAS (sort-and-smoothing) algorithm is consistent. The consistency proof leverages the sparsity concepts from compressed sensing. This algorithm is a very efficient way of estimating graphons, and provide

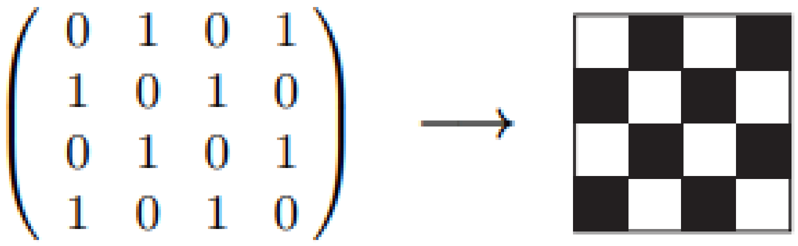


Figure 1: From the adjacency matrix of a labeled graph, the graph's pixel picture is constructed by turning the 1's into black squares

an alternative way of analyzing large-scale network data. Previous methods of graphon estimation algorithms had many restrictions.
For instance, [4] performs graphon estimation conditioned on the node arrangement. When the node arrangement is conditioned on, we can bypass the difficult problem of identifying a canonical representation of the graphon. Other methods estimate canonical graphons. In [5], the authors proposed a method of moments which is theoretically consistent for that purpose. However, the method requires knowledge of all wheels of the network, and hence is computationally infeasible.

The authors of [2] proposes a distance between two networks, that do not have vertex correspondence and could have different number of vertices. The networks are viewed as random samples from (unknown) graphons, and propose a graph distance that estimates the L2-distance between the graphons. Again, the distance is inspired by the sorting-and-smoothing graphon estimator[3]. [2] also present two algorithms for clustering network-valued data based on the proposed graph distance: a distance-based spectral clustering algorithm, and a similarity based semi-definite programming (SDP) approach. Both algorithms derive performance guarantees for under the assumption that the networks are sampled from graphons satisfying certain smoothness conditions. The performance of the algorithm proposed in [2] outperforms other clustering strategies based on graph kernels, graph matching, network statistics etc. on both simulated and real data.

3 Problem Description

Our contribution through this project is to compare the use of graphons for various graph analytic tasks. More rigorously speaking, a graphon [6] is a symmetric measurable function $W : [0,1]^2 \rightarrow [0,1]$. It can be thought as a matrix where each combination of 2 values between $[0,1]$ is assigned to the probability that there might be an edge between those two edges, see Figure 1 1. We evaluate the efficacy of graphon embedding on two fronts: measure of success for graph tasks and time taken to generate the embedding. To compare the results we use graph2vec as a baseline model. The downstream graph analytic tasks that we concerned with are classification and clustering of graph data. We expect that the measure of success for graphon embedding will be slightly worse than graph2vec but we believe that the time taken to construct these embeddings will warrant the use of graphons over graph2vec.

We note that the real - world data is usually heavily imbalanced and to overcome this we extend the use of graphons as data augmenter. For this purpose we implement G-mixup as described by Han et al. [7]. G-mixup builds on the idea of linear interpolation and use points on the constructed line. Thus, we build single graphon for each class and interpolate points between different classes. We then generate samples from the newly constructed graphon.

Table 1: Various graphon functions we used to generate synthetic dataset

ID	w(u, v)
1	u v
2	$\exp\left\{-\left(u^{0.7}+v^{0.7}\right)\right\}$
3	$\frac{1}{4}\left[u^2+v^2+u^{1/2}+v^{1/2}\right]$
4	$\frac{1}{2}(u+v)$
5	$\frac{1}{1+\exp\{-10(u^2+v^2)\}}$
6	$\lvert u-v\rvert$
7	$\frac{1}{1+\exp\{-(\max(u,v)^2+\min(u,v)^4)\}}$
8	$\exp\left\{-\max(u,v)^{3/4}\right\}$
9	$\exp\left\{-\frac{1}{2}\left(\min(u,v)+u^{1/2}+v^{1/2}\right)\right\}$
10	$\log(1+0.5\max(u,v))$

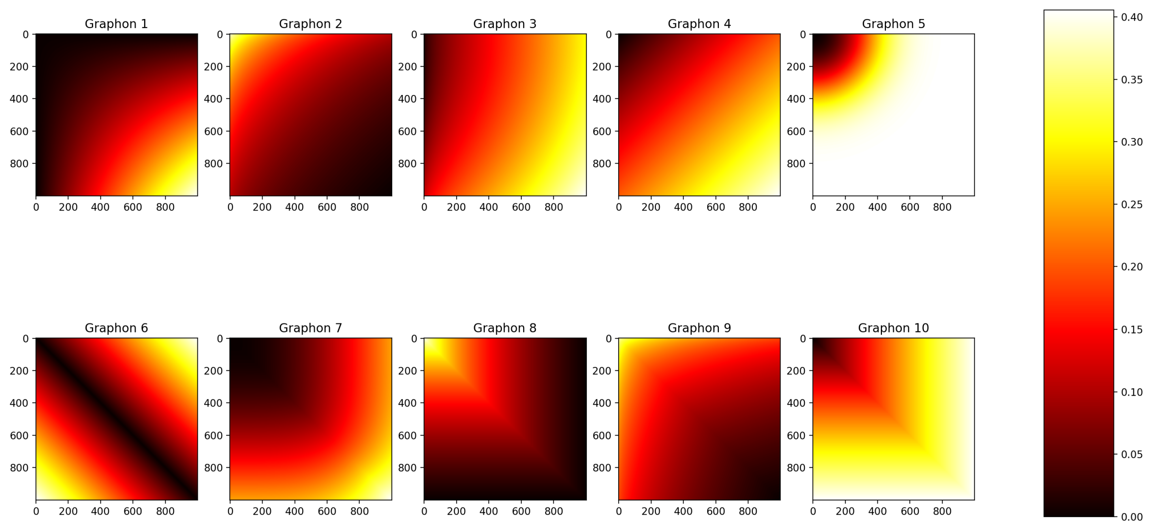


Figure 2: Plots of the graphons as heatmaps. The values represent the probability that an edge is generated between two nodes.

3.1 Data

To perform our analysis we run our experiments on synthetic data to build intuition about the effect of different parameters on the final result. We then use these intuition to perform the graph analytic tasks on real world dataset.

3.1.1 Synthetic dataset

We construct dataset by randomly sampling from known graphons. They are labeled by graphon function used to generate them. The graphon used to generate these samples are provide in Table 1. We visualise the output of the graphons as shown in Figure 2 2. We expect the results to be unequivocally favourable towards graphon based embedding. To counter this we use the help of synthetically generated graphs from other mathematical functions.

3.1.2 Real world dataset

To test the application on real world dataset we take the help of Reddit-dataset [8], Facebook dataset [9], Deezer dataset [10] and Github dataset [10].

4 Methodology and Implementation

Our workflow is as follows: We first load the dataset (sythetic or real), create graphon embeddings, perform a downstream task.

4.1 Loading the dataset

For the synthetic dataset, we sample graphs from a graphon based on equations from Table 1. The way we do this is by creating a set of nodes N $(n_1, \ldots, n_N)$, each sampled from a uniform distribution. For every u, v in this list of nodes, we apply the function of the graphon. The graphon function generates the probability of a weight existing between two nodes. This is how we obtain a graph from a graphon. This process can be repeated multiple times in order to get multiple graphs.

4.2 Creating Graphon embeddings

To generate graphons we use the histogram based approach as described by Chan and Airoldi [11]. The constructed graphons is dependent on the final matrix dimension $N_0 \times N_0$, where N_0 is the dimension of the embeddings. The process of creating graphons is as follows:

Algorithm 1 Algorithm 1 Sort and Smooth (SAS) algorithm

```
        Input: An $n \times n$ graph $G$.
        Output: An $n_0 \times n_0$ estimate of the graphon $\widehat{w}^{est}$.
  Step 1 - Sorting
        Compute empirical degree distribution $d_i = \sum_{j=1}^{n} G_{ij}$
        Sort the degree distribution and determine the corresponding permutation $\widehat{\sigma}$.
        Form a sorted graph $\widehat{A}_{ij} = G_{\widehat{\sigma}(i)\widehat{\sigma}(j)}$.
  Step 2 - Smoothing
        Compute histogram
            $\widehat{H}_{ij} = \frac{1}{h^2} \sum_{i_1=1}^{h} \sum_{j_1=1}^{h} \widehat{A}_{ih+i_1, jh+j_1}$,
        for some bin width $h$.
```

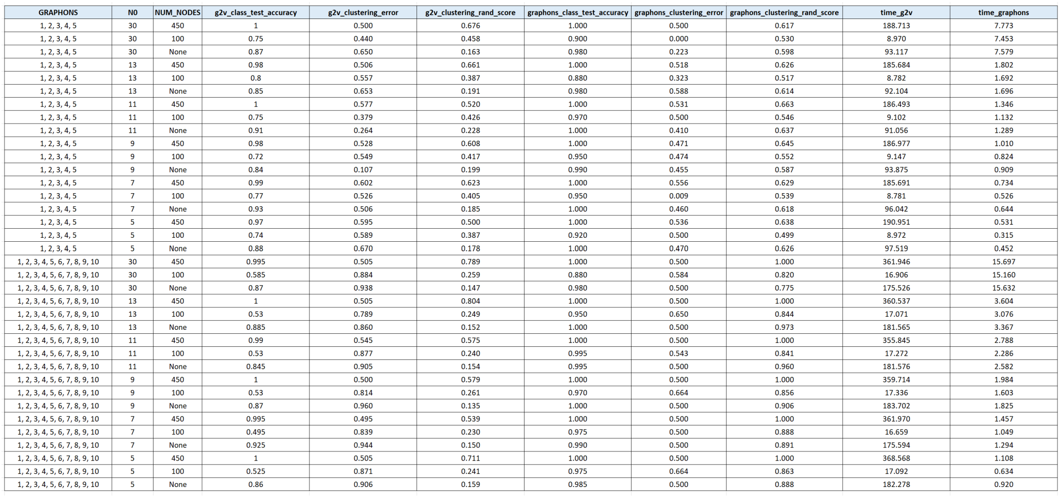
We could also use some variance minimization methods to smoothen the final histogram, but for our experiments, we do not do so. Once we have the graphons, we study the performance of graphons with respect to N_0 by varying it. For graph tasks, we construct the graphons of dimension $N_0 \times N_0$ for each graph G_i in the dataset.

4.3 Downstream tasks

We have two main downstream tasks, mainly classification and clustering. For classification, we consider the unrolled graphon matrix as the embedding. We then find the classification accuracy using a random forest classifier [12]. In this way, we consider the graphons to be an embedding of the original graph. For comparison of performance, we use an already existing implementation of the Graph2Vec algorithm to generate another form of embeddings for the same graph. We then perform classification using two different embeddings and compare their accuracies.
For the clustering task, we consider the graphons as individual points and perform spectral clustering using the distance matrix. The distance matrix whose ith row and jth column is the Frobenius norm defined as $< G_i, G_j >_F = G_i^T G_j$, where G_i, G_j are the histogram approximate of $i^t h$ and $j^t h$ graphon. We perform the same set of experiments but with the baseline Graph2Vec embedding.

5 Experiments

We establish the time taken to generate the graphon by varying N_0 as compared to graph2vec. We present the results in Figures 3, 7, 8. We immediately notice that generating embedding through graphon is faster than graph2vec by 20 times. Next we evaulate using the below graph analytics task



GRAPHONS	N0	NUM_NODES	g2v_class_test_accuracy	g2v_clustering_error	g2v_clustering_rand_score	graphons_class_test_accuracy	graphons_clustering_error	graphons_clustering_rand_score	time_g2v	time_graphons
1, 2, 3, 4, 5	30	450	1	0.500	0.676	1.000	0.500	0.617	188.713	7.773
1, 2, 3, 4, 5	30	100	0.75	0.440	0.458	0.900	0.000	0.530	8.970	7.453
1, 2, 3, 4, 5	30	None	0.87	0.650	0.163	0.980	0.223	0.598	93.117	7.579
1, 2, 3, 4, 5	13	450	0.98	0.506	0.661	1.000	0.518	0.626	185.684	1.802
1, 2, 3, 4, 5	13	100	0.8	0.557	0.387	0.880	0.323	0.517	8.782	1.692
1, 2, 3, 4, 5	13	None	0.85	0.653	0.191	0.980	0.588	0.614	92.104	1.696
1, 2, 3, 4, 5	11	450	1	0.577	0.520	1.000	0.531	0.663	186.493	1.346
1, 2, 3, 4, 5	11	100	0.75	0.379	0.426	0.970	0.500	0.546	9.102	1.132
1, 2, 3, 4, 5	11	None	0.91	0.264	0.228	1.000	0.410	0.637	91.056	1.289
1, 2, 3, 4, 5	9	450	0.98	0.528	0.608	1.000	0.471	0.645	186.977	1.010
1, 2, 3, 4, 5	9	100	0.72	0.549	0.417	0.950	0.474	0.552	9.147	0.824
1, 2, 3, 4, 5	9	None	0.84	0.107	0.199	0.990	0.455	0.587	93.875	0.909
1, 2, 3, 4, 5	7	450	0.99	0.602	0.623	1.000	0.556	0.629	185.691	0.734
1, 2, 3, 4, 5	7	100	0.77	0.526	0.405	0.950	0.009	0.539	8.781	0.526
1, 2, 3, 4, 5	7	None	0.93	0.506	0.185	1.000	0.460	0.618	96.042	0.644
1, 2, 3, 4, 5	5	450	0.97	0.595	0.500	1.000	0.536	0.638	190.951	0.531
1, 2, 3, 4, 5	5	100	0.74	0.589	0.387	0.920	0.500	0.499	8.972	0.315
1, 2, 3, 4, 5	5	None	0.88	0.670	0.178	1.000	0.470	0.626	97.519	0.452
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	30	450	0.995	0.505	0.789	1.000	0.500	1.000	361.946	15.697
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	30	100	0.585	0.884	0.259	0.880	0.584	0.820	16.906	15.160
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	30	None	0.87	0.938	0.147	0.980	0.500	0.775	175.526	15.632
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	13	450	1	0.505	0.804	1.000	0.500	1.000	360.537	3.604
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	13	100	0.53	0.789	0.249	0.950	0.650	0.844	17.071	3.076
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	13	None	0.885	0.860	0.152	1.000	0.500	0.973	181.565	3.367
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	11	450	0.99	0.545	0.575	1.000	0.500	1.000	355.845	2.788
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	11	100	0.53	0.877	0.240	0.995	0.543	0.841	17.272	2.286
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	11	None	0.845	0.905	0.154	0.995	0.500	0.960	181.576	2.582
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	9	450	1	0.500	0.579	1.000	0.500	1.000	359.714	1.984
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	9	100	0.53	0.814	0.261	0.970	0.664	0.856	17.336	1.603
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	9	None	0.87	0.960	0.135	1.000	0.500	0.906	183.702	1.825
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	7	450	0.995	0.495	0.539	1.000	0.500	1.000	361.970	1.457
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	7	100	0.495	0.839	0.230	0.975	0.500	0.888	16.659	1.049
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	7	None	0.925	0.944	0.150	0.990	0.500	0.891	175.594	1.294
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	5	450	1	0.505	0.711	1.000	0.500	1.000	368.568	1.108
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	5	100	0.525	0.871	0.241	0.975	0.664	0.863	17.092	0.634
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	5	None	0.86	0.906	0.159	0.985	0.500	0.888	182.278	0.920

Figure 3: Results of the sweep we performed on synthetic graphs, over N0 and the Number of Nodes (NUM_NODES). To test how the choice of N0 influences the performance of the graphons estimator, as well as making a comparison between the classification and clustering performance of Graph2Vec vs graphons.

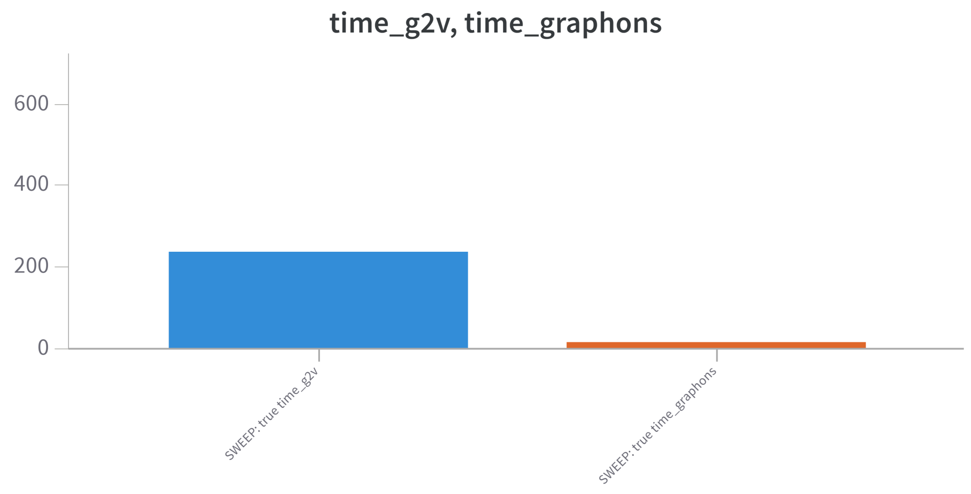


Figure 4: Plots of the time to generate the Graph2Vec and Graphon embeddings

5.1 Clustering and Classification

To evaluate the classification task, we use a Random Forest Classifier with 100 estimators. We compare the train accuracy and test accuracy of these methods for both real world data and synthetic data.

We perform spectral clustering [13] analysis on both methods. To help establish the notion of accuracy, we perform linear sum assignment problem on the true label and predicted cluster group. Thus, the efficacy is presented both in terms of Hungarian score[14] and Random Index score. The former goes from 0 (good) to 1 (bad). The rand Index score can assume a highest value of 1, indicating optimal results for the clustering. As it gets close to zero it indicates that the clustering did not work properly, and negative values can indicate that the model did worse than a random one. Assuming that the number of true classes in the dataset is unknown, we tested how the algorithms performed when using different number of clusters (since that is a necessary parameter for the clustering algorithm we

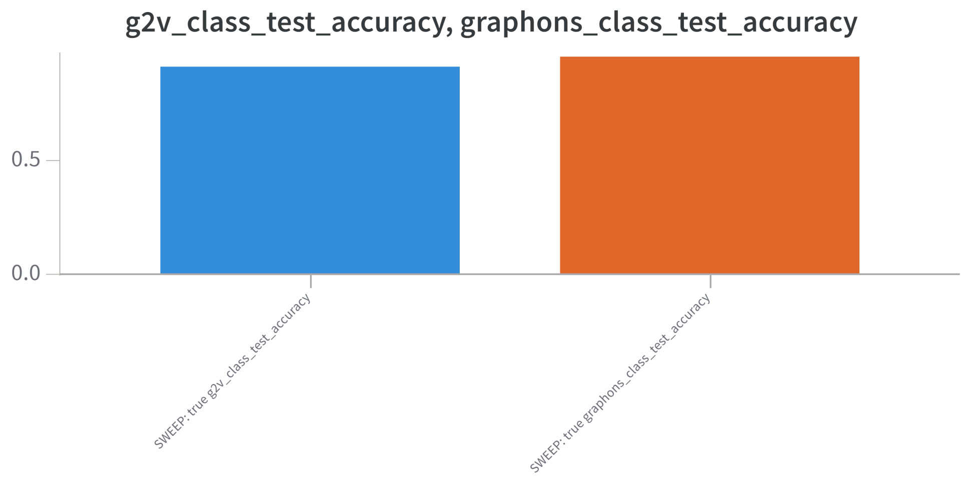


Figure 5: Plots of test classification accuracy for the Graph2Vec and Graphons on Real World data

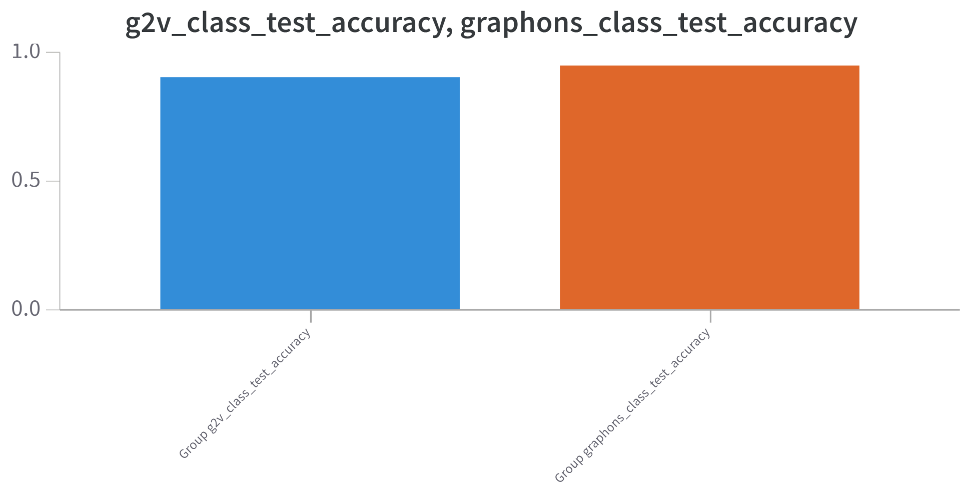


Figure 6: Plots of test classification accuracy for the Graph2Vec and Graphons on synthetic data

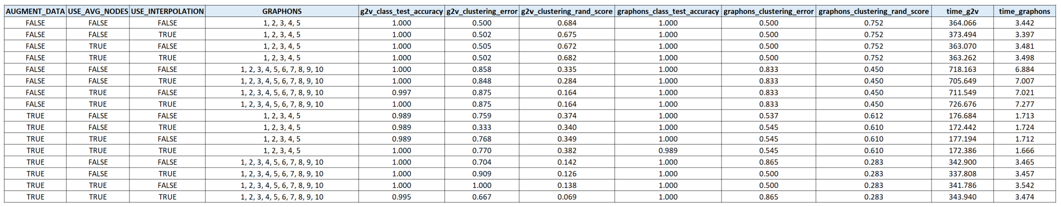
used). Moreover, since we perform spectral clustering, we investigated the consequence of using two or four eigenvectors.

Another set of experiments was instead aimed at understanding how the dimensions of the histogram embedding, governed by the parameter N_0, influences the classification and clustering accuracy with respect to Graph2Vec, when using the synthetic data. We can see the results in table ...

We also explored how the results changed instead when using Real world data.

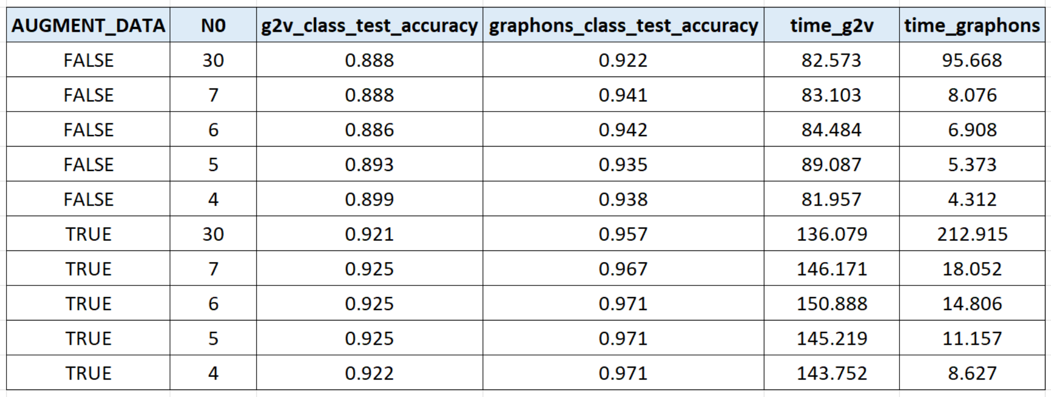
5.2 Data Augmentation

Next we perform augmentation as presented in G-Mixup[7]. We again compare them using real-world data and synthetic data. This is shown in 7 (synthetic data), 8 (real world data). Here we tried different implementations of the Data augmentation. G-Mixup work in the following way, fixing a certain number of nodes N_0 that are desired in the histogram approximation:



AUGMENT_DATA	USE_AVG_NODES	USE_INTERPOLATION	GRAPHONS	g2v_class_test_accuracy	g2v_clustering_error	g2v_clustering_rand_score	graphons_class_test_accuracy	graphons_clustering_error	graphons_clustering_rand_score	time_g2v	time_graphons
FALSE	FALSE	FALSE	1, 2, 3, 4, 5	1.000	0.500	0.684	1.000	0.500	0.752	364.066	3.442
FALSE	FALSE	TRUE	1, 2, 3, 4, 5	1.000	0.502	0.675	1.000	0.500	0.752	373.494	3.397
FALSE	TRUE	FALSE	1, 2, 3, 4, 5	1.000	0.505	0.672	1.000	0.500	0.752	363.070	3.481
FALSE	TRUE	TRUE	1, 2, 3, 4, 5	1.000	0.502	0.682	1.000	0.500	0.752	363.262	3.498
FALSE	FALSE	FALSE	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	1.000	0.858	0.335	1.000	0.833	0.450	718.163	6.884
FALSE	FALSE	TRUE	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	1.000	0.848	0.284	1.000	0.833	0.450	705.649	7.007
FALSE	TRUE	FALSE	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	0.997	0.875	0.164	1.000	0.833	0.450	711.549	7.021
FALSE	TRUE	TRUE	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	1.000	0.875	0.164	1.000	0.833	0.450	726.676	7.277
TRUE	FALSE	FALSE	1, 2, 3, 4, 5	0.989	0.759	0.374	1.000	0.537	0.612	176.684	1.713
TRUE	FALSE	TRUE	1, 2, 3, 4, 5	0.989	0.333	0.340	1.000	0.545	0.610	172.442	1.724
TRUE	TRUE	FALSE	1, 2, 3, 4, 5	0.989	0.768	0.349	1.000	0.545	0.610	177.194	1.712
TRUE	TRUE	TRUE	1, 2, 3, 4, 5	1.000	0.770	0.382	0.989	0.545	0.610	172.386	1.666
TRUE	FALSE	FALSE	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	1.000	0.704	0.142	1.000	0.865	0.283	342.900	3.465
TRUE	FALSE	TRUE	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	1.000	0.909	0.126	1.000	0.500	0.283	337.808	3.457
TRUE	TRUE	FALSE	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	1.000	1.000	0.138	1.000	0.500	0.283	341.786	3.542
TRUE	TRUE	TRUE	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	0.995	0.667	0.069	1.000	0.865	0.283	343.940	3.474

Figure 7: Results of the sweep we performed on synthetic graphs, over N0 and the Number of Nodes (NUM_NODES). The main test is to see the effect of augmentations on the data, along with different methods like interpolation and averaging as mentioned in the experiment section.



AUGMENT_DATA	N0	g2v_class_test_accuracy	graphons_class_test_accuracy	time_g2v	time_graphons
FALSE	30	0.888	0.922	82.573	95.668
FALSE	7	0.888	0.941	83.103	8.076
FALSE	6	0.886	0.942	84.484	6.908
FALSE	5	0.893	0.935	89.087	5.373
FALSE	4	0.899	0.938	81.957	4.312
TRUE	30	0.921	0.957	136.079	212.915
TRUE	7	0.925	0.967	146.171	18.052
TRUE	6	0.925	0.971	150.888	14.806
TRUE	5	0.925	0.971	145.219	11.157
TRUE	4	0.922	0.971	143.752	8.627

Figure 8: Results of the sweep we performed on Real world graphs, over N0 and the Number of Nodes (NUM_NODES). The main test is to see the effect of augmentations on the data, along with different methods like interpolation and averaging as mentioned in the experiment section.

- For each graph in the dataset, obtain an histogram approximation using N_0 nodes
- obtain the "average" graphon for each class, by summing together all the histogram approximations with a specific label and dividing by the number of samples. At this point we obtain a single graphon approximation for each class
- we can use the just found approximation to generate new graphs for the dataset, in the same way graphon functions are used to perform the same operation.

Doing this we can add an arbitrary number of graphs of a specific class to the dataset. However, the only constraint is that the generated graphs can have a number of nodes that is equal or lower than N_0. We believe this could be an issue, since by augmenting the dataset we want new graphs with on average as many nodes as the others. Hence if it is small, then also the generated graphs will be small. This is why we tested different approaches:

- using N_0 as the average number of nodes of the graphs for each class. In this way we can only obtain a histogram approximation of the graphs that have at least as many nodes as N_0, and we don't consider the others.
- use a small value for N_0, so that all the graphs of a class can be used. But then we can use interpolation methods to increase the dimension to the average number of nodes.

6 Results

We used Weights and Biases `https://wandb.ai/site` to run multiple sweeps with various hyperparameters. This works like a massive gridsearch. In terms of speed, graphons are much much faster than Graph2Vec. From Figure 4 we can see the mean run time of all the Graph2Vec run and the Graphon run. Table 3 shows a much more detailed analysis of the times for every run we had.

For the classification accuracy on synthetic data, we see from figure 6 that on average graphons perform as well as graph2Vec. Note that this might be the case as we are generating the graphs from the graphons already, and hence there might be some underlying bias, but we also show that in the real-world dataset, the average accuracy for Graphons are higher than Graph2Vec Figure 5. From the tables you can also

As mentioned above, we use interpolation and averaging in the augmentations part, we notice that averaging performs better than interpolation based on the classification accuracy. Detailed values are mentioned in Table 7.

For clustering, we notice similar results where the graphons score is better than the graph2vec score. We have exactly similar results for the real world dataset as well, and this can be seen in Table 8

7 Summary and Conclusions

From the experiments above, we can confirm that Graphons are a much faster way to generate embeddings when compared to Graph2Vec. The resulting accuracy on a classification task using these embeddings are similar to those from Graph2Vec, and sometimes better. We can also see that selecting the right graphon embedding size is very crucial to estimate its performance.
When performing augmentation, we can see an increase in the accuracy of the classification model.

7.1 Future work

Something we have thought about from the beginning but haven't integrated it with the code is the use of ego-nets. Ego-nets can be used to obtain smaller graphs corresponding to a node, from a large graphs (assuming we take the 1 Hop distance). A interesting extension of this project could be to experiment the influence of graphon embeddings on ego-nets for Node classification.
Another interesting extension could be to see if combining various graphons from egonets would result in a much better graphon estimation of a graph than just computing the graphon of a graph. Ofcourse we could always improve the experiments by running on more graphs of larger node sizes.

8 Who did what

Carlo and Vishal were working on the synthetic data generation and Danny and Simone were working on the real world data methods. All of us were working together on the base code, and each pair was responsible for implementing it for either the real world or synthetic data. Carlo was responsible for setting up the weights and biases for the experiments.
Simone and Danny were working on the augmentation strategies to tackle the unbalanced dataset issue, to make it work on both the synthetic and the real-world data.
All of us together were running different experiments on real-world and synthetic data as we ran multiple sweeps on weights and biases.
We again worked together on the report as we had multiple subsections.

References

[1] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs, 2017. URL `https://arxiv.org/abs/1707.05005`.

[2] Debarghya Ghoshdastidar Mahalakshmi Sabanayagam, Leena C. Vankadara. Graphon based clustering and testing of networks: Algorithms and theory. *ICLR*, 2022.

[3] Edoardo M. Airoldi Stanley H. Chan. A consistent histogram estimator for exchangeable graph models. *ICLR*, 2014.

[4] P. J. Wolfe S. Choi and E. M. Airoldi. Stochastic blockmodels with a growing number of classes. *ICLR*, 2012.

[5] A. Chen J. Bickel and E. Levina. The method of moments and degree distributions for network models. *ICLR*, 2011.

[6] Daniel Glasscock. What is a graphon? 2016. doi: 10.48550/ARXIV.1611.00718. URL `https://arxiv.org/abs/1611.00718`.

[7] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. G-mixup: Graph data augmentation for graph classification. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8230–8248. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/han22c.html`.

[8] Pinar Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1365–1374, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783417. URL `https://doi.org/10.1145/2783258.2783417`.

[9] Lutz Oettershagen, Nils M. Kriege, Christopher Morris, and Petra Mutzel. Temporal graph kernels for classifying dissemination processes, 2019. URL `https://arxiv.org/abs/1911.05496`.

[10] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. Karate club: An api oriented open-source python framework for unsupervised learning on graphs. 2020. doi: 10.48550/ARXIV.2003.04819. URL `https://arxiv.org/abs/2003.04819`.

[11] Stanley Chan and Edoardo Airoldi. A consistent histogram estimator for exchangeable graph models. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 208–216, Bejing, China, 22–24 Jun 2014. PMLR. URL `https://proceedings.mlr.press/v32/chan14.html`.

[12] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[13] Ulrike von Luxburg. A tutorial on spectral clustering. *Stat. Comput.*, 17(4):395–416, 2007. URL `http://dblp.uni-trier.de/db/journals/sac/sac17.html#Luxburg07`.

[14] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: https://doi.org/10.1002/nav.3800020109. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109`.

A Appendix

A.1 Weekly Updates

TASKS 28/09/2022

- Understood the theory of spectral clustering, ran preliminary test on synthetic dataset using Hungarian error measure.
- Structured code to aid ease of experiments.
- Understand results in terms of metric used in paper-2.
- Run the classification on the synthetic data

TASKS 05/10/22

- Plotted graphons.
- Documented the code.
- Added Wandb for efficiently storing classification and clustering scores

TASKS 14/10/22

- Found the “data leakage” during the graph2vec embedding process.
- Ran a sweep for classification on Synth Data.
- Read the metrics for clustering.
- Run a bigger sweep on the server.
- Run classification on Real data.
- Run the sweep for clustering (locally).
- Maybe start with the data augmentation paper.

TASKS 19/10/22

- Tested out the code on real-world data.
- Due to class imbalance, we decided to use the augmentation paper to augment the data of each class.
- Run the code on the cluster for synthetic data.
- Complete the augmentation part.
- Run the experiments on the real-world data on the cluster.

TASKS 2/11/22

- Wrote the code for augmentation and tested to see if it runs.
- Started writing the report.
- Ran the biggg sweep (some errors due to RAM).
- Continue testing using the sweeps for results.

TASKS 9/11/22

- Tried to do likelihood but didn’t work, creating a graphon for each class (SVD)
  - Using the cross product and summing up all values
  - Performing the Frobenius norm
- Run sweep with D.A. on Synth
  - Using n graphs or n/2 + n/2 augmented
  - Using interpolation to increase the dimensions or the average number of nodes
- Added subsampling for Real graphs Data augmentation

- Continue testing using the sweeps, for real data.

TASKS 16/11/22

- Started to optimize / clean the code repository.
- Run sweep for synthetic data.
- Added these latter results to the report.

TASKS 23/11/22

- Finished to optimize / clean the code repository.
- Run the last sweep for augmented data and gathered all results.
- Added the results to the report.

TASKS 30/11/22

- Reflected on every result that we obtained during the project.
- Finilized the report.

TASKS 07/11/22

- We had a call with the opposing group. Each group briefly presented their project, tlaked about their experience and the respective company they were working with.
- A lot of things weren't clear about each other's group since it the arguments were new for the opponents. However, we took the time to discuss and clirify those concepts that were unclear.

A.2 How much time each of us spent on this project.

Everyone worked approximately 10 hours a week.