Recognizing Power-law Graphs by Machine Learning Algorithms using a Reduced Set of Structural Features

Alane Marie de Lima[1], **André Luis Vignatti**[1], **Murilo Vicente Gonçalves da Silva**[1]

[1]Department of Computer Science – Federal University of Paraná (UFPR)
Caixa Postal 19.031 – 81531-980 – Curitiba – PR – Brazil

{amlima,vignatti,murilo}@inf.ufpr.br

Abstract. *The empirical study of large real-world networks in the last 20 years showed that a variety of real-world graphs are power-law. There are evidence that optimization problems seem easier in these graphs; however, for a given graph, classifying it as power-law is a problem in itself. In this work, we propose using machine learning algorithms (KNN, SVM, Gradient Boosting and Random Forests) for this task. We suggest a graph representation based on [Canning et al. 2018], but using a much simplified set of structural properties of the graph. We compare the proposed representation with the one generated by the* graph2vec *framework. The experiments attained high accuracy, indicating that a reduced set of structural graph properties is enough for the presented problem.*

1. Introduction

Large real-world graphs, also commonly called complex networks, occur in applications based on natural and social phenomena. Many of these networks have a power-law distribution in their vertex degree sequence, which can be informally described as a function that decreases exponentially in x as x grows large for fixed exponent $k > 0$ and a proportionality constant α, i.e, $f(x) = \alpha x^{-k}$. These networks, called power-law graphs in this paper, have few vertices with large degree, and many others vertices with small degree (Figure 1). The empirical study of large real-world networks in the late 1990's and early 2000's [Faloutsos et al. 1999,Barabasi and Albert 1999,Kleinberg et al. 1999,Broder et al. 2000,Kumar et al. 2000,Kleinberg and Lawrence 2001,Guelzim et al. 2002,Siganos et al. 2003, Eubank et al. 2004] showed that a number of real-world graphs are power-law.

From a computer science perspective, a key point is that there is evidence that optimization problems seem easier in power-law graphs compared to general graphs. In particular, some problems from this class have the expected factor of approximation reduced when the instance is a power-law graph [Vignatti and da Silva 2016, Silva et al. 2013, Gast and Hauptmann 2014, Gast et al. 2015, Gast et al. 2012]. However, the task of classifying whether a graph is power-law before submitting it as input to an optimization problem requires some effort.

One common and simple way to know if some sequence follows a power-law distribution consists on the log-log plot test, i.e., setting both the horizontal and vertical axis to logarithmic scales. A power-law distribution appears as a straight line in this new plotting, since the function that describes this distribution turns to a linear equation in log-log plot (Figure 2). The main disadvantage of this approach, though, is the fact that other distributions (such as the log-normal) also appear as a straight line in this test.

Figure 1. Example of a power-law graph [Grandjean 2014].

The work of [Clauset et al. 2009] consists in a statistical method to quantify the power-law behavior into a given distribution. The authors conclude that for many of the cases of the given datasets (both synthetic and real-world applications), a power-law distribution turns out to be a reasonable description. However, for some of these cases, log-normals and stretched exponential distributions are also compatible. In other cases, such as the distribution of earthquakes, for example, the power-law hypothesis is compatible if an exponential cutoff in the extreme tail of the distribution is assumed.

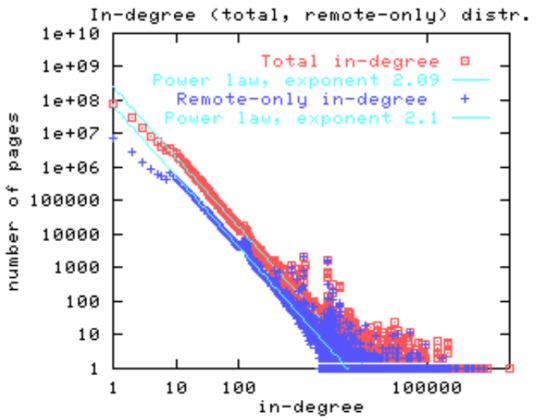

Figure 2. A log-log plot for the power-law distribution of a graph that represents the web, where the vertices are the pages and the edges are the links between them [Broder et al. 2000].

In this work, we propose machine learning techniques to distinguish power-law graphs among real and synthetic complex networks, including data with log-normal distribution. The field of machine learning looks to develop algorithms that learn from data, and such algorithms are used in many applications including natural language processing, handwritting recognition, and graph classification, what is relevant to our work (in Section 2, we present some recent development in this topic) [Mitchell 1997].

We consider using structural properties related to the degree of the vertices of the given graph, as suggested by [Canning et al. 2018], to represent the feature vector. In

order to compare our proposed approach against approaches that require a more complex graph representation, we use the *graph2vec* framework [Narayanan et al. 2017] to generate feature vectors for the graphs used in the experiment. This framework is a recent tool for graph classification by machine learning algorithms that allows us to apply general purpose classification algorithms to data. Our main contribution is showing that a much simplified set of structural properties of a graph are enough for the desired classification procedure, even on hard instances.

The text is organized as follows. Section 2 presents related work on power-law recognition in data and graph classification by machine learning algorithms. Section 3 describes some concepts and definitions used in this work. Section 4 describes the dataset and the feature extraction configuration. Section 5 discusses the results and finally, Section 6 contains the conclusion.

2. Related Work

This Section is divided in two parts: (1) recognition of power-law distribution in data and (2) graph classification by machine learning algorithms.

The most known approaches for recognizing power-law distributions in data are the log-log plot test and the statistical framework from [Clauset et al. 2009]. As already mentioned, a power-law distribution appears as a straight line in a log-log plot. Despite its simplicity, the condition of being straight in a log-log plot is necessary but not sufficient, since other distributions such as the log-normal also appear as straigth.

The statistical method proposed by [Clauset et al. 2009], which uses hypothesis test, not only fits a given distribution into a power-law model, but also quantifies the power-law behavior by estimating its parameters. The authors addressed, though, the difficulty of distinguishing a power-law distribution from a log-normal one, since some datasets well fit in both distributions. They highlight the fact that extremely large datasets are required in these cases.

On the other hand, when it comes to general graph classification problem, many recent machine learning approaches have emerged. A recent survey from [Ghosh et al. 2018] describes graph kernels. These methods measure the similarity between pairs of graphs by decomposing each graph into its atomic substructures, such as random walks, paths and rooted trees. Graph kernels allow kernel-based algorithms such as support vector machines (SVM) to work directly on graphs without feature extraction, but they become unusable for other general purpose learning algorithms. Hence, neural networks approaches [Bonner et al. 2016, Tixier et al. 2017, Pham et al. 2017] have been proposed as a way around.

A recent work from [Canning et al. 2018] represents graphs as arrays where their features form a combination of structural properties, e.g., density; maximum, minimum and average degree; assortativity coefficient; total, maximum and average triangles; average and global clustering coefficient; maximum k-core and a lower bound for the maximum clique. The authors conclude that real complex networks have distinct structural properties, which ensure machine learning algorithms to have high accuracy in classification and clustering problems on these graphs, even if simple features are used to represent each graph. They also conclude that is possible to classify with high accuracy not only whether a graph is synthetic or not, but also the network model that generates it.

From the aforementioned work, though, no results about the power-law graph classification problem by machine learning algorithms have been found. This work, so, intends to investigate the performance of this type of algorithms applied to the referred problem, especially in the case where data contains graphs with log-normal distribution.

3. Theoretical Foundation

Sections 3.1, 3.2 and 3.4 describes the definitions used in this work. Section 3.3 introduces *graph2vec*, a neural network based framework that generates the feature vector of a graph.

3.1. Distributions

We use the following definitions for the power-law and log-normal distributions.

Definition 1 *A function that satisfies*

$$f(x) = \alpha x^{-k}$$

is said to follow a power-law distribution, for constant values $k > 0$ *and* $\alpha > 0$.

Definition 2 *A continuous probability distribution of a non-negative random variable* X *follows a log-normal distribution if its logarithm is normally distributed, that is, for* $Y = \ln X$,

$$g(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(y-\mu)^2/2\sigma^2}$$

where μ *is the mean,* σ *is the standard deviation and* $-\infty < y < \infty$ *(Figure 3).*

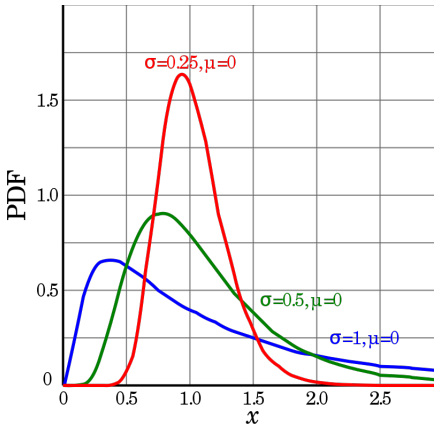

Figure 3. Probability density function of some log-normal distributions with differing σ.

3.2. Graph Definitions

In this Section, we define the structural properties used in our proposed feature vectors.

Let $G = (V(G), E(G))$ be a graph where $V(G)$ is a finite set (the vertices of G) and $E(G) \subseteq \binom{V(G)}{2}$ (the edges of G). Two vertices u and v are neighbors in G if $\{u, v\} \in E(G)$. The degree of a vertex v in G is the number of neighbors of v in G and is denoted by $d_G(v)$. The maximum and minimum degrees of a vertex in G are denoted by $\Delta(G)$ and $\delta(G)$, respectively. The average degree of the vertices in G is defined as

$d_{avg} = \frac{1}{|V(G)|} \sum_{v \in V(G)} d_G(v)$. The density of a graph is the ratio of the edges in the graph to the amount of all possible edges, that is, $|E(G)|/|\binom{V(G)}{2}|$.

The induced subgraph of G by a set $S \subseteq V(G)$ is denoted by $G[S]$. For a vertex v, let $X \subseteq V(G)$ be a set where every $y \in V(G[X])$ is reachable from v with distance up to d, for $d \in \mathbb{N}$. Then $G[X]$ denotes a rooted subgraph of G of degree d.

3.3. The graph2vec Framework

This framework is a recent unsupervised representation learning technique for graphs proposed by [Narayanan et al. 2017]. It consists of a neural network embedding that learns representations of arbitrary sized graphs. It is based on the well-known *doc2vec* framework [Le and Mikolov 2014] that is used in natural language processing for documents representation learning.

The main idea of the *graph2vec* framework is to generate similar representations to similar instances by finding all rooted subgraphs around each vertex (up to a certain degree) of the graphs in the whole dataset. Hence, graphs that have close subgraphs on their structures will have close embeddings. More details can be checked in [Narayanan et al. 2017].

3.4. Feature Normalization

The values of each feature vector are rescaled to the [0,1] range, since the graphs in database have different ranges in the number of vertices. We use min-max normalization, where for each n-dimensional array x and for all $i \in [1, n]$, the normalized array x' is obtained as follows:

$$x'[i] = \frac{x[i] - min(x)}{max(x) - min(x)}$$

Similar results were observed for other normalizations.

4. Experiments

The algorithms are implemented in Python 2.7 using the NetworkX[1] and the *scikit-learn*[2] libraries. All experiments were performed on a GNU/Linux (Ubuntu 14.04) with the following hardware configurations: (i) Processor: Intel(R) Xeon E5 v3 (Haswell) 2.50 GHz, x86_64 architecture, 4 cores and 8 vCPUs. (ii) Cache memory: 32KB x 4 L1 Instruction cache; 32KB x 4 L1 Data cache; 256KB x 4 L2 cache; 30MB L3 cache. (iii) RAM: 52GB.

We describe our dataset structure and the feature extraction approaches used in this work.

4.1. Dataset

The dataset contains synthetic and real-world graphs from different sizes, applications and classes, totalizing 3145 instances, where 1236 of them are power-law. All synthetic graphs were artificially generated by computational models available in NetworkX library. The artificial power-law instances were obtained by the models proposed by [Barabasi

[1]https://networkx.github.io/

[2]https://scikit-learn.org

and Albert 1999], [Holme and Kim 2002] and [Bollobás et al. 2003], which simulate power-law complex networks. The remaining synthetic instances include graphs from different classes that do not follow a power-law on their vertex degree distribution, such as complete, grid and log-normal graphs.

The real-world instances used in this work, which model instances from the real-world, as the name itself suggests, are available in the Network Repository[3] and the Stanford Large Network Dataset Collection[4].

Table 1 summarizes the structure of the dataset, where the graphs highlighted in italics have power-law degree distribution. Since the generated log-normal distributions have decimal values, then each value was rounded up to its closest integer. So, graphs with these distributions are approximately log-normal.

Table 1. Dataset structure

	Description	# of graphs	highest # of vertices
Synthetic graphs	*Barabasi-Albert* [Barabasi and Albert 1999]	599	10000
	Holme-Kin [Holme and Kim 2002]	300	100000
	Directed scale-free [Bollobás et al. 2003]	200	10000
	General graphs	1003	10000
	Log-normal graphs	300	10000
Real-world networks	*CAIDA AS Relationships Datasets* [Leskovec et al. 2005]	122	26475
	Gnutella peer-to-peer networks [Foster et al. 2002]	9	10876
	EuroSis web-mapping [5]	1	1285
	Enzymes [Narayanan et al. 2017]	600	95
	Brain networks [Rossi and Ahmed 2015]	6	2000
	Physics Collaboration Networks [Leskovec et al. 2005]	3	18772
	Amazon Products Co-purchasing [Leskovec et al. 2007]	1	403394
	EU Email Communication Network [Leskovec et al. 2005]	1	265214

4.2. Feature Extraction

We represent each graph as a feature vector of some of its structural properties, based in the idea of [Canning et al. 2018]. We call our proposed feature vector as *struct_vector*, which has the following properties: a) maximum degree, b) minimum degree, c) density, d) average degree, e) number of vertices that have the maximum degree and f) number of vertices that have the minimum degree. The chosen properties are related to the degrees of the vertices, as the degree distribution of a graph is a sufficient condition to classify it as power-law. In power-law graphs, this kind of representation will set a small value for feature e), a large value for feature f) and a small value for feature c), since graphs from this class are sparse.

To compare our representation with another one which uses more complex features of the graph, we generate graph embeddings by the *graph2vec* framework. As we already mentioned, this framework consists on a deep neural network that creates similar representations to graphs that have similar rooted subgraphs on its structure. Each graph embedding in our experiments is a 128-dimension array.

[3]http://networkrepository.com/

[4]https://snap.stanford.edu/data/

4.3. Classification Algorithms

We compare prediction results for the following algorithms implemented in *scikit-learn* library: k-nearest neighbors (KNN), support vector machine (SVM), gradient boosting and random forest, since these algorithms have been used by previous work in graph classification by machine learning algorithms. We use hold-out validation, and we divide the dataset separating 2/3 of it for training and the remaining part for testing. The graphs selected to compose the training set are selected uniformly at random. The instances of each set are shuffled.

5. Results

In Table 2, we report the best parameters[6] found by grid search that lead to the results with higher accuracy. Although our database is balanced in respect to the amount of power-law and non-power-law instances, we use precision, recall and f1-score metrics to measure the occurrences of false positives, that is, the non-power-law graphs that were incorrectly classified as power-law. We report the results for these metrics in Table 3. Class 0 and Class 1 denote negative (not a power-law graph) and positive (power-law graph), respectively.

Table 2. Best parameters for the classifiers

Algorithm	Graph representation	Best parameters	Accuracy
KNN	*struct_vector*	neighbors = 3	99.34%
	graph2vec		90%
SVM	*struct_vector*	C = 1000 $\gamma = 1$	99.53%
	graph2vec	C = 10 $\gamma = 0.1$	99.14%
Gradient Boosting	*struct_vector*	estimators = 50 learning rate = 0.1 max. depth = 3	99.91%
	graph2vec	estimators = 100 learning rate = 0.5 max. depth = 3	97.05%
Random Forest	*struct_vector*	estimators = 100 max. depth = 5	99.91%
	graph2vec	estimators = 50 max. depth = 9	96.86%

We achieve high accuracy in both representations and all the classification algorithms, although the *struct_vector* was slightly better to this problem. We highlight that while our approach generates the feature vectors in approximately 10 minutes, the *graph2vec* framework performed the graph feature extraction in 5398 minutes (which corresponds to approximately 3.75 days).

The ensemble algorithms, i.e., gradient boosting and random forests, got the best performance in the classification task. It is aligned with the state-of-the-art, since the work of [Canning et al. 2018] already achieved good performance on graph classification

[6]For KNN, we set number of neighbors=$\{3, 5, 11\}$. For SVM, we set $C = \{1, 10, 100, 1000\}$ and $\gamma = \{0.001, 0.01, 0.1, 1\}$. For the other classifiers, the parameters configuration are: number of estimators=$\{50, 100, 200\}$, learning rate=$\{0.01, 0.1, 0.5\}$ and maximum depth=$\{3, 5, 9, 10\}$.

Table 3. Classifiers analysis

Algorithm	Graph representation	Precision	Recall	F1-Score
KNN	*struct_vector*	Class 0 = 99.2% Class 1 = 99.5%	Class 0 = 99.7% Class 1 = 98.7	Class 0 = 99.8% Class 1 = 99.4%
	graph2vec	Class 0 = 85.8% Class 1 = 100%	Class 0 = 100% Class 1 = 74.4%	Class 0 = 92.4% Class 1 = 85.3%
SVM	*struct_vector*	Class 0 = 99.5% Class 1 = 99.5%	Class 0 = 99.7% Class 1 = 99.3%	Class 0 = 99.8% Class 1 = 99.6%
	graph2vec	Class 0 = 98.9% Class 1 = 99.5%	Class 0 = 99.7% Class 1 = 98.3%	Class 0 = 99.3% Class 1 = 98.9%
Gradient Boosting	*struct_vector*	Class 0 = 100% Class 1 = 99.8%	Class 0 = 99.8% Class 1 = 100%	Class 0 = 99.9% Class 1 = 99.9%
	graph2vec	Class 0 = 95.6% Class 1 = 99.5%	Class 0 = 99.7% Class 1 = 92.3%	Class 0 = 97.6% Class 1 = 96.08%
Random Forest	*struct_vector*	Class 0 = 100% Class 1 = 99.8%	Class 0 = 99.8% Class 1 = 100%	Class 0 = 99.9% Class 1 = 99.9%
	graph2vec	Class 0 = 95.1% Class 1 = 100%	Class 0 = 100% Class 1 = 91.9%	Class 0 = 97.5% Class 1 = 95.8%

using simple arrays with few features. In our problem, this result is convenient, since it is not necessary to have information about more complex structures of a graph to classify it as power-law.

The inferior performance of *graph2vec* to this task can be explained by the fact that the generated rooted subgraphs of power-law graphs can be very similar to the ones of other sparse graphs, like the cycle and regular graphs. Although getting slightly smaller accuracy results to our problem and being more complex, *graph2vec* seems to be a good and competitive alternative to the other representation. Even though knowing the sub-structures of graph is not necessary to our problem, the performance of the framework in this task indicates that it can be used for other graph classification tasks. This also answers the question on how power-law graphs are structurally distinct from the other graphs when information different from the degree distribution are considered. That is, graphs from diverse domains and applications have distinct structural properties that make data distinguishable, as [Canning et al. 2018] have already stated.

Surprinsigly, for both types of representation, the errors in prediction results most happened as false negatives (as Table 3 shows in precision and recall values). This seems interesting because the main challenge for power-law recognition is the opposite, that is, the occurrence of false positives.

In fact, the proposed graph representations and the applied algorithms perform well to distinct power-law graphs from the approximately log-normal ones. In our experiments, only two occurrences of false positives to a graph with this distribution were found, both for the *struct_vector* representation. One of them arised in KNN algorithm, and the other arised in SVM algorithm. Tables 4 and 5 show the confusion matrix of the testing step for the aforementioned cases.

It indicates that using machine learning algorithms for recognizing power-law graphs as well as this kind of distribution in other types of data is a powerful and fast alternative in relation to the other approaches.

Table 4. Confusion matrix of the KNN algorithm using the *struct_vector* representation

		Predicted	
		Non-power-law	Power-law
Actual	Non-power-law	641	2
	Power-law	5	401

Table 5. Confusion matrix of the SVM algorithm using the *struct_vector* representation

		Predicted	
		Non-power-law	Power-law
Actual	Non-power-law	641	2
	Power-law	3	403

6. Conclusion

In this work, the main question to be answered was how accurate a power-law graph can be predicted by machine learning algorithms using either simple features of a graph or a more complex approach such as *graph2vec*. In conclusion, both alternatives give good results, although the simpler approach is sufficient to the presented problem. We address the fact that although the work of [Canning et al. 2018] is comparable to our work, they have not reported results for the task of classification of power-law graphs. Besides, even though they have used only structural properties that can be efficient calculated, our approach uses features that can be obtained by looking once at the vertex degree distribution of a graph. In the work of [Canning et al. 2018], to get the structural property of the maximum k-core, for example, is necessary to process each vertex neighborhood more than once.

In comparison to the statistical framework of [Clauset et al. 2009], our approach does not quantify the scaling and the constant factor of the power-law behavior. However, for some applications, it may be enough having only the information that a distribution follows a power-law or not. An advantage of using our approach lies in the fact that it can distinguish well an approximately log-normal distribution from a power-law one without having an extremely large dataset, a difficulty addressed by [Clauset et al. 2009] in their work.

For further research, one can consider analyzing the performance of machine learning algorithms to distinguish power-law graphs from instances that follow power-law variants in their vertex degree distribution, such as the power law with exponential cutoff, curved power-law and broken power-law, since our work did not include these types of instance in the database.

Additionally, one can consider the creation of an end-to-end classifier for power-law graphs, in which the embedding is learned alongside the classification, since we focused on comparing different graph representations for classifiers from the literature in this work.

References

Barabasi, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Bollobás, B., Borgs, C., Chayes, J., and Riordan, O. (2003). Directed scale-free graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 132–139, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Bonner, S., Brennan, J., Theodoropoulos, G., Kureshi, I., and McGough, A. S. (2016). Deep topology classification: A new approach for massive graph classification. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3290–3297.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. (2000). Graph structure in the web. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 33(1-6):309–320.

Canning, J. P., Ingram, E. E., Nowak-Wolff, S., Ortiz, A. M., Ahmed, N. K., Rossi, R. A., Schmitt, K. R. B., and Soundarajan, S. (2018). Predicting Graph Categories from Structural Properties. *arXiv e-prints*, page arXiv:1805.02682.

Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2009). Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703.

Eubank, S., Kumar, V., Marathe, M. V., Srinivasan, A., and Wang, N. (2004). Structural and algorithmic aspects of massive social networks. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 718–727. Society for Industrial and Applied Mathematics.

Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On power-law relationships of the internet topology. *ACM SIGCOMM Computer Communication Review*, 29(4):251–262.

Foster, I., Ripeanu, M., and Iamnitchi, A. (2002). Mapping the gnutella network. *IEEE Internet Computing*, 6:50–57.

Gast, M. and Hauptmann, M. (2014). Approximability of the vertex cover problem in power-law graphs. *Theoretical Computer Science*, 516:60–70.

Gast, M., Hauptmann, M., and Karpinski, M. (2012). Improved approximation lower bounds for vertex cover on power law graphs and some generalizations. *arXiv preprint arXiv:1210.2698*.

Gast, M., Hauptmann, M., and Karpinski, M. (2015). Inapproximability of dominating set on power law graphs. *Theoretical Computer Science*, 562:436–452.

Ghosh, S., Das, N., Gonçalves, T., Quaresma, P., and Kundu, M. (2018). The journey of graph kernels through two decades. *Computer Science Review*, 27:88 – 111.

Grandjean, M. (2014). La connaissance est un réseau. *Les cahiers du numérique*, 10(3):37–54.

Guelzim, N., Bottani, S., Bourgine, P., and Képès, F. (2002). Topological and causal structure of the yeast transcriptional regulatory network. *Nature Genetics*, 31(1):60.

Holme, P. and Kim, B. J. (2002). Growing scale-free networks with tunable clustering. *Physical Review E*, 65(2):026107.

Kleinberg, J. and Lawrence, S. (2001). The structure of the web. *Science*, 294(5548):1849–1850.

Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. S. (1999). The web as a graph: Measurements, models, and methods. In *Proceedings of the 5th Annual International Conference on Computing and Combinatorics*, COCOON'99, pages 1–17, Berlin, Heidelberg. Springer-Verlag.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., and Upfal, E. (2000). Stochastic models for the web graph. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 57–65. IEEE.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Leskovec, J., Adamic, L. A., and Huberman, B. A. (2007). The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1).

Leskovec, J., Kleinberg, J., and Faloutsos, C. (2005). Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 177–187, New York, NY, USA. ACM.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. (2017). graph2vec: Learning distributed representations of graphs. *CoRR*, abs/1707.05005.

Pham, T., Tran, T., Dam, H., and Venkatesh, S. (2017). Graph classification via deep learning with virtual nodes. *CoRR*, abs/1708.04357.

Rossi, R. A. and Ahmed, N. K. (2015). The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Siganos, G., Faloutsos, M., Faloutsos, P., and Faloutsos, C. (2003). Power laws and the as-level internet topology. *IEEE/ACM Transactions on Networking*, 11(4):514–524.

Silva, M. O. d., Gimenez-Lugo, G. A., and Da Silva, M. V. (2013). Vertex cover in complex networks. *International Journal of Modern Physics C (IJMPC)*, 24(11):1–9.

Tixier, A. J., Nikolentzos, G., Meladianos, P., and Vazirgiannis, M. (2017). Classifying graphs as images with convolutional neural networks. *CoRR*, abs/1708.02218.

Vignatti, A. L. and da Silva, M. V. G. (2016). Minimum vertex cover in generalized random graphs with power law degree distribution. *Theoretical Computer Science*, 647:101–111.