k-simplex2vec: a simplicial extension of node2vec

Celia Hacker
Departement of Mathematics
EPFL
Lausanne, Switzerland
celia.hacker@epfl.ch

Abstract

We present a novel method[1] of associating Euclidean features to simplicial complexes, providing a way to use them as input to statistical and machine learning tools. This method extends the node2vec algorithm to simplices of higher dimensions, providing insight into the structure of a simplicial complex, or into the higher-order interactions in a graph.

1 Introduction

In many areas of research, data comes with its own intrinsic structure. That data might be non-Euclidean, leading in some cases to difficulties in using familiar tools from statistics and machine learning. To address this problem, some methods provide ways to learn how to understand this data in a Euclidean space. It is important to impose some constraints on the Euclidean representation of the data to best render its initial structure. In other words, we look for a function $F : X \to \mathbb{R}^d$ that preserves some of the structure encoded in X and where d is small relative to the size of X. One may hope that the image of F retains enough of the original structure of X that it can be discerned using statistical or machine learning tools. This structure-preserving map is often called an *embedding*, and $\mathbb{R}^d$ the *feature space*. The well known word2vec [7, 8], and node2vec [6] algorithms are examples of such methods, the former preserving some form of structure in a text, and the latter preserving that of a graph. The idea behind node2vec, which is itself based on word2vec, is to do random walks starting at each vertex of the graph, recording the co-occurrences of nodes in random walks. One then maximizes a function of $\langle F(u), F(v)\rangle$ for vertices u, v that appear together in the same random walk, where $\langle \cdot, \cdot \rangle$ denotes the scalar product in the feature space.

The aim of this paper is to generalize node2vec to a wider class of objects, namely to *simplicial complexes*. These objects are a higher dimensional analog of graphs, which in addition to nodes (0-simplices) and edges (1-simplices), also contain triangles (2-simplices) and k-simplices, which are formed by $k+1$ vertices. In the world of topological data analysis these naturally encode higher-order relations in data and posses a more complex topological structure than graphs [4]. As a consequence, more structure is taken into consideration while defining random walks on simplicial complexes.

The first step towards generalizing node2vec is to define a notion of random walk on simplicial complexes. In the graph case the walker can go from one vertex to another whenever they are connected by an edge. For simplices of higher dimension, there are different possibilities of generalization. As stated in [10], a direct generalization of the walk on a graph can be made by connecting any two k-simplices if they are a face of the same $(k+1)$-simplex. However, the generalization of the graph Laplacian to simplicial complexes, in [9], encodes connections in terms of both upper connections through cofaces and lower connections through faces. In what follows we use walks inspired by [9], using both connections. The relationship between random walks and simplicial

[1] The code is available at https://github.com/celiahacker/k-simplex2vec

Topological Data Analysis and Beyond Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Laplacians, although not as straightforward as in the graph case [3], reveals interesting connections to the algebraic topology of the simplicial complex, as shown in [10, 9].

A generalization of node2vec was proposed in [2], in terms of random walks on the Hasse diagram of a simplicial complex, which is a graph encoding face relations. The method then simply applies node2vec to that graph. We believe this approach does not have a strong link to the algebraic topological properties of the simplicial complex, by the nature of the walks being used. In [2], only the effect of the algorithm on the nodes is studied, omitting the representations of higher dimensional simplices present in the complex. Other examples of applications of random walks on simplicial complexes can be found in [13].

2 Proposed Method

A simplicial complex X is a collection of finite sets closed under taking subsets. The building blocks are k-simplices, which are sets of cardinality $k+1$. We denote the subset of k-simplices of X by X_k. The local structure of the simplicial complex is encoded in the two types of connections between simplices. Walking on the simplices through upper and lower connections gives a way to transcribe the local structures of the simplicial complex into point clouds using the process described below and in Algortihm 1. The behaviour of the random walks on the k-simplices is encoded in a transition matrix P, which is a square matrix of size $|X_k|$, with each row summing to 1. The row P_σ, corresponding to a simplex σ, encodes the probability that a walker at σ has of jumping to any k-simplex in the complex. As mentioned in the introduction, many choices can be made when constructing P, yielding many variations of the random walks. Here, we choose to assign equal probability to each neighbor of a simplex, independently of whether it is an upper or lower neighbor. For this algorithm we perform N random walks starting at each simplex $\sigma \in X_k$, which we truncate at a fixed length l. We wish to obtain a representation $F : X_k \to \mathbb{R}^d$ mapping the k-simplices into a space of dimension d, usually small relative to the size of X_k. To obtain this mapping F, we move $F(\sigma)$ and $F(\tau)$ closer whenever σ, τ often appear in the same random walks, while on the contrary moving them further apart if they do not appear together in the walks. This optimization is translated into a maximum likelihood problem, using the random walks as similarity measure [11].

We want to maximize the probability of observing a certain random walk starting at a simplex σ conditioned on its representation, over possible mappings $F : X_k \to \mathbb{R}^d$, i.e.,

$$\max_F \sum_{\sigma \in X_k} \log(\Pr(\mathrm{RW}(\sigma)|F(\sigma))),$$

where $\mathrm{RW}(\sigma)$ are the simplices reached by a random walk at σ. Assuming independence and symmetry in the feature space, the conditional probability is

$$\Pr(\mathrm{RW}(\sigma)|F(\sigma)) = \prod_{\tau \in \mathrm{RW}(\sigma)} \Pr(\tau|F(\sigma)),$$

where $\Pr(\tau|F(\sigma)) = \frac{\exp\langle F(\tau), F(\sigma)\rangle}{\sum_{\nu \in X_k} \exp\langle F(\nu), F(\sigma)\rangle}$. This leads to maximizing the following objective function

$$\max_F \sum_{\sigma \in X_k} \left[-\log\left(\sum_{\nu \in X_k} \langle F(\sigma), F(\nu)\rangle \right) + \sum_{\tau \in \mathrm{RW}(\sigma)} \langle F(\tau), F(\sigma)\rangle \right]$$

using stochastic gradient descent. For more details we refer the reader to [6, 11]. We point out that when $k = 0$, we recover the node2vec algorithm.

3 Experimental results

In this section we provide experimental results for the algorithm described above. We have explored several data sets yielding interesting results, however in this paper we choose to focus on one particularly well-understood model of synthetic data, in order to capture the essence of the method.

Methodology. We run the algorithm above ten times on a simplicial complex built on a stochastic block model (SBM) [1]. The model consists of three blocks of 20 vertices, with intra-block probability

Algorithm 1 k-simplex2vec

k-simplex2vec (simplicial complex X, dimension of simplices k, dimension of feature space d, walks per simplex N, walk length l)
  $X_k = k$-simplices of X
  P = matrix of transition probabilities
  Walks = [ ]
  for i = 1 to N **do**
    for all $\sigma \in X_k$ **do**
      walk = **SimplicialWalks**(σ, P, X_k)
      append walk to Walks
    end for
  end for
  F = StochasticGradientDescent(X_k, d, Walks)
  return F

SimplicialWalks (start simplex σ, length l, transition matrix P)
  walk = $[\sigma]$
  for $j = 1$ to l **do**
    choose τ from neighbours(σ) according to P_σ
    append τ to walk
  end for
  return walk

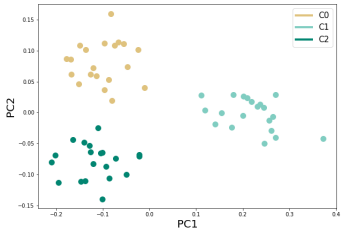


Figure 1: The PCA projection of the node2vec representation of the stochastic block model shows three cluster corresponding to the three groups of nodes.

0.8 and inter-block probability 0.3. We build a simplicial complex out of the SBM by taking its clique complex, where every complete subgraph on $k+1$ vertices gives rise to a k-simplex. The complex contains 60 nodes, 809 edges and 3610 triangles. We run the algorithm above for $k = 1, 2$, i.e., on the edges and the triangles of the clique complex, yielding ten point clouds representing the 1-simplices and ten point clouds representing the 2-simplices. On the nodes of the graph, the algorithm is equivalent to node2vec. In this case the result is discussed in [5] and is shown in Figure 1, where the clusters perfectly recover the three groups of vertices of the original SBM. This result can be extended in the following way to simplices of higher dimension.

The 1-simplices and 2-simplices can be sorted into different classes. There are two types of 1-simplices, those that have both nodes within one block and those with nodes in two different blocks, yielding a total of six classes, depending in which blocks the vertices are. In the same way there are three sorts of triangles in the clique complex. We distinguish 2-simplices that have all vertices inside the same block, 2-simplices that have two vertices in one block and the third in another, and finally 2-simplices with exactly one vertex in each block. These yield ten classes of 2-simplices, depending on the blocks the vertices belong to. The types of simplices are shown in Figures 5 and 6.

Various parameters of the algorithm can be tuned, i.e., the length of the walks, the number of walks starting at each simplex, and the dimension of the feature space, which is in general much less than $|X_k|$. We vary these parameters within a certain range. For instance, the number of walks and their length vary between 10 and 50 for the edges and between 20 and 60 for the 2-simplices. We consider feature spaces of dimension 10, 20, and 30. After running the algorithm on the edges and triangles,

Table 1: Rand index for varying number of walks and dimension of the feature space of edges.

	Number of walks				
dimensions	10	20	30	40	50
10	0.98 ± 0.01	0.98 ± 0.00	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01
20	0.98 ± 0.01	0.98 ± 0.01	0.99 ± 0.00	0.98 ± 0.00	0.99 ± 0.00
30	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.04	0.99 ± 0.00

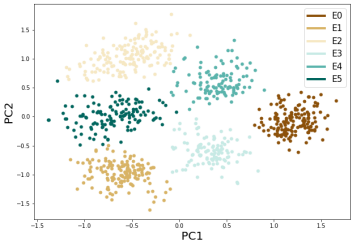
Table 2: Rand index for varying number of walks and dimension of the feature space of triangles.

	Number of walks				
dimensions	20	30	40	50	60
10	0.81 ± 0.10	0.82 ± 0.10	0.81 ± 0.10	0.82 ± 0.10	0.82 ± 0.10
20	0.88 ± 0.10	0.88 ± 0.10	0.88 ± 0.10	0.88 ± 0.10	0.88 ± 0.10
30	0.88 ± 0.10	0.88 ± 0.10	0.88 ± 0.10	0.88 ± 0.10	0.88 ± 0.10

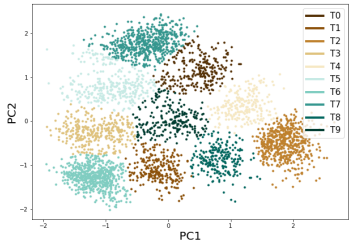
we apply a clustering algorithm to the resulting point clouds and compare the clusters with the known types of k-simplices using the Rand index [12]. The Rand index compares two label assignments to the same set of points, measuring the proportion of points with the same labels in both cases.

Results. We perform clustering on the point clouds using the kmeans algorithm. The results of the clustering for the edges and triangles are plotted in Figure 2 using PCA in two dimensions for a given set of parameters. We run the clustering with $k = 6$ for the edges and $k = 10$ for the 2-simplices to reflect the types of 1 and 2- simplices present in the clique complex of the SBM. We study the clusters obtained on the point clouds by comparing them to the ground truth given by the types of simplices discussed above. The resulting Rand indices over ten trials of the algorithm are presented in Tables 1 and 2, for a fixed walk length of 20. We consider feature spaces of dimensions 10 , 20, 30 and vary the number of walks starting at each simplex. Results using the DBSCAN algorithm for clustering instead of kmeans can be found in Section A.2.

The six classes of edges are recovered with an accuracy above 0.97, and the ten classes of triangles with a Rand index above 0.80, reaching 0.88 in the feature space of dimension 30. As we can see in this range of parameters, varying the number of walks for a fixed length does not change the accuracy. Rather, the dimension of the feature space influences the accuracy of the clustering. In this range of parameters the higher the dimension, the greater the precision. As illustrated in these results, this method allows us to distinguish, with high precision, among different classes of simplices in the clique complex of a graph, directly generalizing the result for node2vec on the vertices. Through these results, we gain insight into the higher dimensional structure of the underlying graph, highlighting the many types of interactions between groups of vertices present in the graph.



(a) Clusters of edges from the feature space of dimension 20, using a walk length of 20 and 40 walks per edge.



(b) Clusters of triangles from the feature space of dimension 20, using a walk length of 20 and 40 walks per triangle.

Figure 2: Both figures were obtained from the features representing the 1 and 2-simplices by applying PCA. The colors reflect the clusters determined by the kmeans algorithm.

4 Conclusion

The results presented here are promising. We have obtained similiar results for different types of network data, such as other random graph models, social networks and neural data. For the sake of brevity and clarity of the message, these were not included in the paper. Understanding the results on a graph with well known structure is key to understanding how the algorithm works and provides intuition for more complicated data sets.

So far we have considered the algorithm only on simplicial complexes obtained from network structured data. We seek to extend the use of this method to simplicial complexes that are not necessarily clique complexes of graphs. Further work includes a thorough study of the effect of changing each hyperparameter of the algorithm, as well as studying the effect of different types of walks, as mentioned in Section 2. After further investigation of the results using this algorithm we will study the algorithmic aspects such as the complexity and performance of the method.

In analogy to the graph case where the random walks detect the connected components, i.e., corresponding to homology in dimension 0, the random walks on simplicial complexes as described in [9] and [10] have an interesting connection to global algebraic topology properties of the simplicial complex. We believe that with the method described in this paper, we can extract other topologically interesting features by means of random walks on simplicial complexes.

Acknowledgments and Disclosure of Funding

I thank Kathryn Hess for guidance and advice throughout this project, as well as Gard Spreemann and Stefania Ebli for the useful discussions. The author was supported by the NCCR Synapsy grant of the Swiss National Science Foundation.

References

[1] Emmanuel Abbe. Community detection and stochastic block models: Recent developments. *Journal of Machine Learning Research*, 18(177):1–86, 2018.

[2] Jacob Charles Wright Billings, Mirko Hu, Giulia Lerda, Alexey N Medvedev, Francesco Mottes, Adrian Onicas, Andrea Santoro, and Giovanni Petri. Simplex2vec embeddings for community detection in simplicial complexes, 2019.

[3] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[4] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. 01 2010.

[5] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78 – 94, 2018.

[6] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[7] Tomas Mikolov, G.s Corrado, Kai Chen, and Jeffrey Dean. Efficient estimation of word representations in vector space. pages 1–12, 01 2013.

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[9] Sayan Mukherjee and John Steenbergen. Random walks on simplicial complexes and harmonics. *Random Structures & Algorithms*, 49(2):379–405, 2016.

[10] Ori Parzanchevski and Ron Rosenthal. Simplicial complexes: Spectrum, homology and random walks. *Random Structures & Algorithms*, 50(2):225–261, 2017.

[11] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.

[12] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[13] Michael T. Schaub, Austin R. Benson, Paul Horn, Gabor Lippner, and Ali Jadbabaie. Random walks on simplicial complexes and the normalized hodge 1-laplacian. *SIAM Review*, 62(2):353–391, 2020.

A Supplementary Materials

A.1 Random walks on simplicial complexes

In this section we illustrate the upper and lower connections that appear in simplicial complexes. As in Section 2, one can connect two simplices of same dimension by a connection through either a common face or a common coface. In either case the difference in dimension is 1.

Figure 3 illustrates the walks using upper connections only: a walk from one edge to another, through common triangles. In contrast, Figure 4 shows that simplex σ and ν are connected through a walk using only connections through shared 0-simplices. One can consider these two types of walks separately, or use them together, mixing both types, as is done in Algorithm 1.

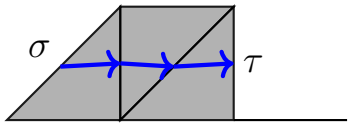


Figure 3: A walk from σ to τ using only upper connections, i.e., through common triangles.

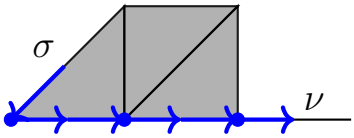


Figure 4: A walk from σ to ν using only lower connections, namely through vertices.

A.2 Results

In this section we complete the results shown in Section 3. The first additions are Figures 5 and 6, both depicting an example of a stochastic block model with three communities. In Figure 5 edges are highlighted by colors corresponding to the clusters shown in Figure 2a, distinguishing between edges within a block or between two blocks. In Figure 6, three types of triangles are shown, corresponding to three of the clusters in Figure 2b, which distinguish the triangles into ten classes.

In the results of Section 3, we show only the clustering using kmeans, which relies on some form of knowledge of the data. We present here some results using the DBSCAN algorithm, which clusters points based on density.

Figure 7 shows a clustering on the feature space of the edges, using DBSCAN. The algorithm does not cluster all points, some remain unclustered. The proportion of unclustered points is on average 5% for the feature space of the edges. We discard those points when computing the Rand index for the resulting clusters. Figure 7 shows a clustering using DBSCAN on the feature space of the triangles. In this case, the average proportion of unclustered points is 15%.

The Rand indices over ten trials are shown in Tables 3 and 4. Again, the length of the walks is fixed to be 20, and we vary the dimension and number of walks just as in the results presented in Section 3.

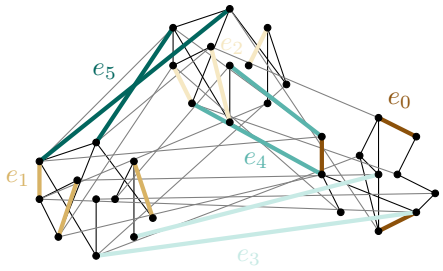


Figure 5: The edges are colored corresponding to the clusters they belong to in Figure 2a, namely an edge e_i corresponds to a point in cluster E_i.

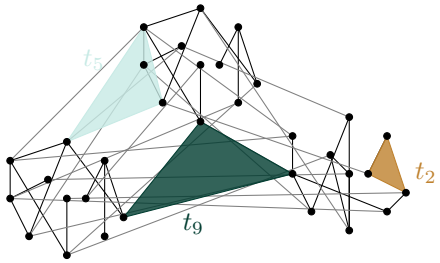


Figure 6: The three triangles depict some of the types of triangles distinguished by the clusters of Figure 2b. Triangle t_2, with all three vertices inside the same block corresponds to a point in cluster T_2. Triangle t_5, with two vertices in one block and the third in another one, illustrates the points in cluster T_5. Finally, triangle t_9 illustrates the 2-simplices which have exactly one vertex in each block, corresponding to points in cluster T_9.

In this case the Rand indices obtained using the DBSCAN algorithm are approximately the same as those obtained using kmeans.

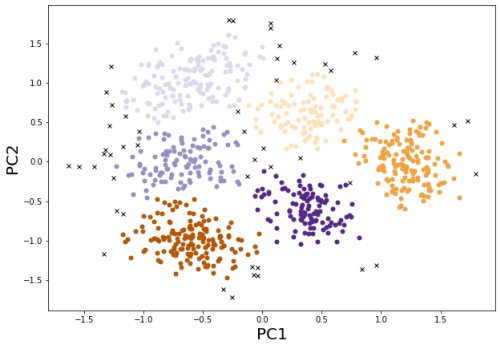


Figure 7: The plot shows the PCA in two dimensions of the feature space of dimension 20, length of walks 10 and 10 walks per edge. The points marked by crosses correspond to those that were not clustered by the DBSCAN algorithm.

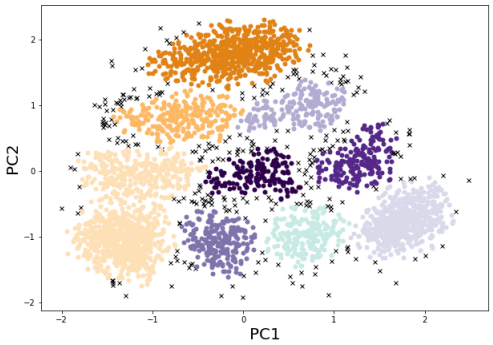


Figure 8: The plot shows the PCA in two dimensions of the feature space of dimension 20 for the edges, with length of walks 10 and 10 walks per 2-simplex. The points marked by crosses correspond to those that were not clustered by the DBSCAN algorithm.

Table 3: Rand index for varying number of walks and dimension of the feature space of edges, with fixed walk length 20.

	Number of walks				
dimensions	10	20	30	40	50
10	0.98 ± 0.05	0.98 ± 0.05	0.99 ± 0.00	0.98 ± 0.05	0.98 ± 0.05
20	0.96 ± 0.03	0.96 ± 0.15	0.98 ± 0.04	0.99 ± 0.00	0.99 ± 0.05
30	0.89 ± 0.10	0.96 ± 0.15	0.95 ± 0.05	0.98 ± 0.04	0.95 ± 0.15

Table 4: Rand index for varying number of walks and dimension of the feature space of triangles, with fixed walk length 20.

	Number of walks				
dimensions	20	30	40	50	60
10	0.83 ± 0.12	0.81 ± 0.15	0.83 ± 0.15	0.83 ± 0.15	0.81 ± 0.13
20	0.85 ± 0.11	0.84 ± 0.11	0.84 ± 0.12	0.85 ± 0.11	0.85 ± 0.11
30	0.80 ± 0.12	0.79 ± 0.16	0.80 ± 0.12	0.80 ± 0.14	0.79 ± 0.15