CS224W Project Report

Deep Learning with K-Means Applied to Community Detection in Networks

Alexandre Vilcek
vilcek@gmail.com

Abstract

Identifying communities, or clusters, in graphs is a task of great importance when analyzing network structures. A telecommunications provider, for instance, would like to identify communities of customers that place a large amount of calls to each other, in order to create more effective, directed marketing campaigns. Another example would be a financial institution, trying to identify and understand communities of customers that have a high amount of financial transactions between each other. There is also a wide applicability of community detection in life sciences research as well: for example, when studying protein-protein interaction networks.

There is a wide range of algorithms and methods that can be applied, in order to identify communities in network structures, such as Spectral Clustering, Modularity Maximization, and Hierarchical Clustering. Spectral Clustering is one of the traditional algorithms that is suitable for network clustering because it first map the original data points into a lower-dimensional space, where the clustering properties of the graph tend to be much more evident, allowing for a subsequent application of standard clustering techniques such as K-Means. Despite its good results, Spectral Clustering presents some computational challenges when applied to very large networks.

Recent research shows that techniques and algorithms from Deep Learning, such as layered neural networks-based auto-encoders, are suitable for the task of mapping data points into lower-dimensional spaces, which can be useful for a variety of further tasks such as data clustering and classification.

There is also recent research showing that a K-Means based approach is also suitable for a lower-dimensional data representation that can replace traditional neural networks-based auto-encoders in a Deep Learning pipeline.

In this project we will create a new processing pipeline for non-overlapping community detection in network structures based entirely on K-Means. We will show that this approach is similar to a traditional Deep Learning auto-encoder in its ability to learn useful representations of the original data in a lower-dimensional space, making the data clustering task easier to accomplish. We will then test its applicability for the specific challenges of community detection in networks and compare its performance with the traditional Spectral Clustering approach.

1. Introduction and Background

1.1 Community Detection in Networks

There is not a universal, unique definition of what a community is in the context of network analysis. A very comprehensive study on this topic was performed in [8]. The exact definition will depend on the specific system and/or application considered. Nevertheless, a general intuition is that a community is constituted by a group of cohesive nodes, in the sense that there is much more links inside that cohesive group than links connecting that group to nodes outside it.

Consider for example the network depicted by the graph shown in Fig. 1a: we define a graph $G = (V, E)$ as a set of vertices (or nodes) V and a set of edges (or links) E. We would like to find one or more communities $C = (V_c, E_c)$, defined as a subset of G. We then consider the internal and external degrees of a node $v \in V_c$ as k_v^i and k_v^e, respectively. According to the previous intuition, a good community would be a sub graph C, such that for each node $v \in V_c$ we maximize k_v^i and minimize k_v^e. This can be seen in Fig. 1b, where edges linking nodes inside C are drawn in black and edges linking nodes in C to nodes outside C are drawn in red.

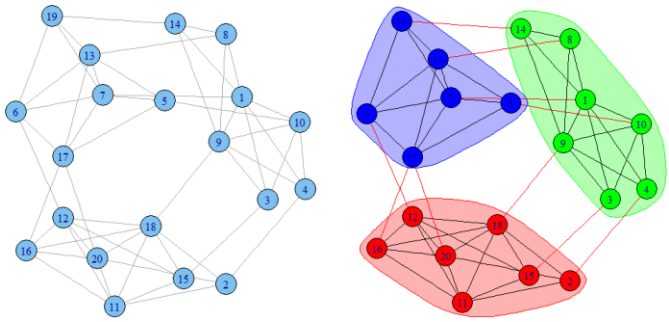

Fig.1a (left): *a graph G with 20 nodes and 52 edges*
Fig. 1b (right): *three communities (color-shadowed in red, green, and blue) defined for the graph G*

1.2 Spectral Clustering

Spectral Clustering [8],[14],[18],[20] is one of the traditional and popular methods of data clustering that is well suited for the problem of finding communities in networks.

When applying Spectral Clustering to an arbitrary dataset, the idea is to represent that dataset by its similarity matrix (or a derivation of it).

Consider for example the graph $G = (V, E)$ of Fig. 1. Then a similarity matrix would be constructed by applying a pairwise similarity function S to every pair of nodes $(v_i, v_j) \in V$ with S being symmetric and non-negative, i.e. $S(v_i, v_j) = S(v_j, v_i) \geq 0, \forall i, j = 1, .. |V|$.

In practice, what Spectral Clustering does is to partition the data according to a lower-dimensional representation that is obtained from computing k eigenvectors of the graph Laplacian matrix L obtained from that data, being k the number of desired clusters. When computing eigenvectors, we consider those corresponding to the k smallest non-zero eigenvalues of L. After embedding the original data into a lower-dimensional representation, the final clustering result is obtained by running K-Means on that representation.

In Fig. 2 we see a plotting of the top-2 eigenvectors whose eigenvalues are non-zero, where the three communities found by K-Means are highlighted in different colors and point shapes. This corresponds to the graph depicted in Fig. 1. It is easy to see that performing clustering on that new representation of the graph becomes a trivial task.

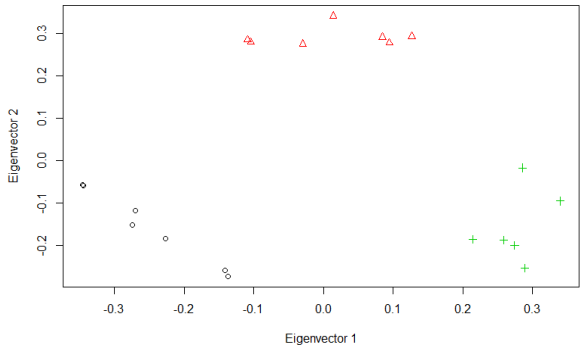

Fig. 2: *three communities found by K-Means applied to the top-2 eigenvectors of L; here $L = D - A$, where D is the diagonal matrix formed by the node degrees of G, and A is the adjacency matrix of G**

It may seem non intuitive, at first glance, why not applying K-Means directly on the original data. It turns out that standard K-Means is capable of separate data only linearly in an n-dimensional space. On the other hand, by embedding the data from an n-dimensional space into a new k-dimensional space, where $k \ll n$, Spectral Clustering makes it possible to linearly separate the data. In Fig. 3a and Fig. 3b we see a simple example showing that: what we actually want to separate are two 'spirals' developing in opposite directions, as correctly identified by Spectral Clustering.

** We note here that for a trivial graph like in this example, the node degree is sufficient as a similarity function. But in practice, for larger and real-word networks, we need a proper similarity function that is able to capture local similar structures in the network, such as Jaccard, Gaussian, or Cosine Similarity.*

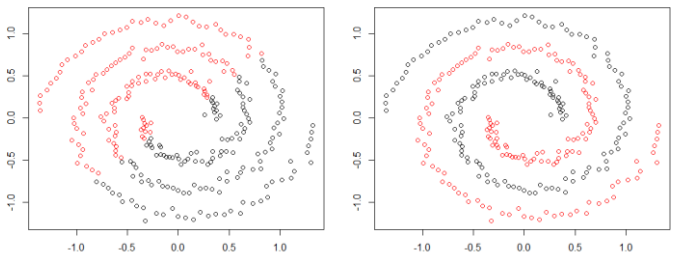

Fig.3a (left): *K-Means fails to identify the two desired 'spirals'*

Fig. 3b(right): *Spectral Clustering correctly identifies two 'spirals', depicted in red and black*

1.3 Deep Learning and Auto-Encoders

In the area of Artificial Intelligence and Machine Learning, In recent years, lots of research has been developed around deep architectures that allow for higher level abstractions for representing and understanding data [2],[7],[11],[19]. Usually these architectures are comprised of artificial neural networks or similar constructs that aim to mimic the layered-processing of information that occurs in the human brain (particularly in the neocortex region). This approach has yielded very good results particularly in tasks related to computer vision and natural language processing.

There is a lot of variety in the possible architectures and algorithms related to Deep Learning. In the context of this study, we will focus on some of the characteristics of Deep Learning that are relevant for the proposed method:

- Use many layers of non-linear processing units for extraction and embedding of features.
- Is based on unsupervised learning.
- Allows for a distributed representation of the data, in the sense that the underlying data can be explained by a set of hierarchical factors.

In the context of Deep Learning, an auto-encoder is the unit responsible for the extraction and embedding of features in an (usually) unsupervised fashion. Given an input $x \in \mathbb{R}^n$ from the original data, the auto-encoder tries to learn an encoding function $f(x) \in \mathbb{R}^k, k \ll n$. It does so, by iteratively minimizing the error of reconstructing x through a decoding function $g(f(x)) \cong x \in \mathbb{R}^n$ (see Fig. 4).

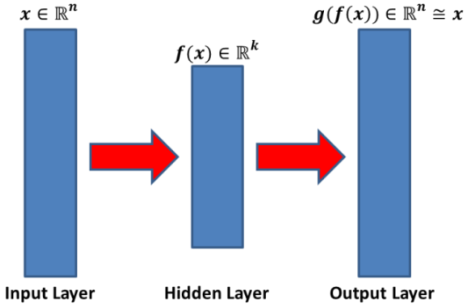

Fig. 4: *a high level representation of an auto-encoder based on a single-layer neural network*

The embedding given by an auto-encoder can then be used to better representing the data for the final learning task, for example a classification or clustering.

In [16] the authors show that Spectral Clustering can be viewed as the minimization of the reconstruction error of the graph similarity matrix S, in a way much similar to an auto-encoder. In the case of Spectral Clustering, the embedding is given by the top $k-1$ eigenvectors of the matrix L, as exemplified in 1.2.

It turns out that, building an embedding hierarchy learned by recursively applying $f(x) \in \mathbb{R}^k$, one is capable of extracting a

distributed representation of the data that tends to better represent the original data as the number of embedding layers in the architecture increase. This behavior was consistently proved in data classification tasks such as described in [6] and similar researches.

1.4 K-Means

In the fields of Machine Learning and Data Mining, perhaps K-Means is the most known and studied method for clustering analysis [12].

Standard K-Means works as following: consider the data to be clustered $X = \{x \in \mathbb{R}^n\}$ and $C = \{c_j, j = 1,..k\}$ the set of k clusters to group X on. K-Means will try then to minimize a loss function $\mathcal{L}$ that measures the squared error between the computed mean of each cluster c_j and its assigned cluster members $\{x_i\}$, for all clusters being considered:

$$\mathcal{L} = \sum_{j=1}^{k} \sum_{x_i \in c_j} \left|\left| x_i - \mu_j \right|\right|^2$$

In this project, we will explore a characteristic of K-Means that is very similar to what we discussed in 1.3: the ability to learn higher level representations of the data in a lower-dimensional space, as shown in [5],[6].

It turns out that, as we will show empirically, for the problem of community detection in networks we only need a very simple encoding schema for the embedding of the original data. This is sometimes called hard-assignment or Boolean encoding and can be formulated by:

$$f(x_i) \in \mathbb{R}^k = \begin{cases} \mu_j, if\ x_i \subset c_j \\ 0, otherwise \end{cases}$$

2. Previous Work

There is a lot of previous research that investigated Spectral Clustering or derived approaches both applied to general clustering problems and specifically to community detection in networks [8],[14],[18],[20].

It is a known fact that the two most expensive computational steps in the Spectral Clustering algorithm are the eigenvalue decomposition of the graph Laplacian matrix, which has time complexity of $O(n^3)$, and the computation of the graph similarity matrix, which has time complexity of $O(n^2)$, both for a graph with n nodes. This makes the algorithm unsuitable for analyzing large network structures. Several approaches that try to address those limitations were investigated:

In [4], the authors investigate techniques for "sparsification" or approximation calculation of the similarity matrix, so that optimized eigensolvers that can be easily parallelized can be used.

In [16], as previously described in 1.3, the authors successfully replace the expensive eigensolver calculation in the Spectral Clustering by a stacked, neural networks-based auto-encoder.

In [17], the authors propose an implementation of Spectral Clustering based on the MapReduce paradigm for parallel, distributed computations. As in [16], they explore the sparsity characteristic of the graph similarity matrix, and optimized eigensolvers.

3. Proposed Approach

3.1 Motivation

In this project we propose a novel approach to perform Graph Clustering for Network Community Detection that is entirely based on K-Means. K-Means is well known for its computational simplicity, ease of implementation, and ease of scalability in parallel distributed computing frameworks.

We have seen the potential computational complexity and scalability issues of Spectral Clustering when applied to large Graphs. We have also seen in [16] a novel approach that seems to outperform Spectral Clustering, yet reducing some of those complexity and scalability issues. On the other hand, we know that the approach proposed in [16], which is based on techniques of feature-learning through auto-encoders implemented with neural networks, is non-trivial to implement for getting good results [2],[5].[11].

Here we propose to replace the eigenvector decomposition step in Spectral Clustering, as defined in [14], by an multi-layer auto-encoder pipeline implemented using only the K-Means clustering algorithm in a recursive way. We call this algorithm Deep K-Means.

3.2 Proposed Implementation for Deep K-Means

Here we detail the proposed algorithm step by step:

a. We begin with a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges.

b. Compute a pairwise similarity matrix S from G, using a similarity function $F_S(v_i, v_j);\ v_i, v_j \in V; i, j = 1..|V|$

c. Compute a diagonal matrix D, whose elements $d_i; i = 1..|V|$ are the node degrees for nodes $v_i \in V; i = 1..|V|$

d. Define a K-Means auto-encoder module K_A comprised by the K-Means computation and hard-assignment as defined in 1.4.

e. We define the initial input data as the matrix $X = D^{-1}S$, the desired number of layers of K_A as l, the desired number of clusters for each layer of K_A as c, and the final desired number of communities as k.

f. For each layer $l_1,\ l_2, \dots, l_m$, we compute $X_l = K_{A,l}(X_{l-1}, c_l)$, such that $c_l = \frac{\#\ of\ columns\ of\ X_{l-1}}{2}$

g. As the last step, we apply standard K-Means to X_{l_m} to find the desired k communities of G.

The embedding of the data matrix into lower-dimensional spaces has the analogous effect of mapping it to its top eigenvectors. Fig. 5 below show a plotting of a 100-node

network with 5 communities, after embedding its initial matrix X (as defined in e. above) into a 4-layer Deep-KMeans. For the sake of visualization, only 2 of the 6 final features are represented.

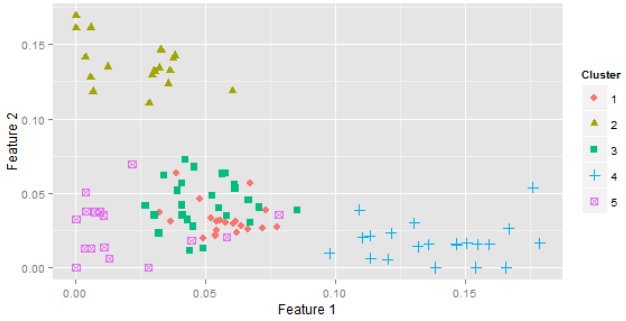

Fig. 5: *visualization of a 100x100 initial matrix after running a 4-layer Deep KMeans and embedding it into a 100x6 matrix (showing only 2 of 6 features)*

3.3 Experiments

We performed a series of experiments to evaluate the accuracy, computational time complexity, and scalability of Deep K-Means, comparing it with an implementation of Spectral Clustering as defined in [14], on the task of finding communities in network structures.

For those experiments, we analyzed network data with ground-truth communities, both synthetic and real-world data.

To generate synthetic data, we used the approach proposed in [13], where the generated data incorporates important features of real networks for the problem of community detection, like the heterogeneity of node degree distributions and community sizes. Table Tab. 1 below summarizes some of the main characteristics of the tested networks.

We also ran experiments using the Football dataset, as described in [22].

In all experiments, the accuracy obtained was evaluated using the Normalized Mutual Information NMI between the

communities detected by the algorithm C_1 and the corresponding ground-truth communities C_2, according to:

$$NMI(C_1, C_2) = \frac{2\, I(C_1, C_2)}{H(C_1) + H(C_2)}$$

Where: $I(C_1, C_2)$ is the Mutual Information between C_1 and C_2, that represents the amount of shared information between C_1 and C_2 and is given by:

$$I(C_1, C_2) = \sum_{x \in C_1} \sum_{y \in C_2} p(x,y) \log\left(\frac{p(x,y)}{p_1(x)p_2(y)}\right)$$

And $H(C)$ is the Entropy of C, that represents the information contained in C and is given by:

$$\mathrm{H(C)} = \sum_{\mathrm{x \in C}} \mathrm{p(x)log(p(x))}$$

A vector C defines a community distribution for each node in the graph.

graph id	*nodes*	*edges*	*max degree*	*min degree*	*degree distrib*	*diameter*	*avg path length*
1	100	502	27	5	power law	4	2.4
2	200	1548	49	7	power law	4	2.4
3	300	2398	50	7	power law	4	2.6
4	400	3022	50	7	power law	5	2.7
5	500	3632	50	7	power law	4	2.8
6	600	4530	50	7	power law	5	2.9
7	700	5399	50	7	power law	5	2.9
8	800	6060	50	7	power law	5	3.0
9	900	6894	50	7	power law	5	3.1
10	1000	7736	50	7	power law	5	3.1
11	2000	15206	50	7	power law	6	3.4
12	3000	28980	75	8	power law	5	3.3
13	4000	39572	80	8	power law	5	3.3
14	5000	49758	80	8	power law	5	3.4

graph id	*largest clique*	*communities*	*largest community*	*smallest community*
1	9	5	25	16
2	13	6	48	22
3	20	10	44	21
4	13	13	45	24
5	13	16	43	21
6	22	22	41	20
7	16	24	41	21
8	16	27	43	21
9	18	31	42	20
10	12	30	49	20
11	15	59	49	20
12	37	76	62	26
13	23	75	80	30
14	21	100	80	30

Tab. 1: *main characteristics of the synthetic networks*

3.3.1 Experiment #1

In this experiment we ran both Deep K-Means and Spectral Clustering on the synthetic networks described earlier. The goal of this experiment is to investigate if the proposed algorithm can provide better clustering performance than Spectral Clustering in a consistent way.

In order to maximize the accuracy of Deep K-Means, we tuned the auto-encoder pipeline with the following parameters

- Number of layers: 3
- Maximum number of iterations for K-Means on each layer: 25
- Number of random initializations for K-Means on each layer: 10
- Similarity function: Jaccard

We ran both Deep K-Means and Spectral Clustering 10 times for each network and computed the average of the clustering accuracy as the normalized mutual information compared to the ground-truth communities.

3.3.2 Experiment #2

In this experiment we analyze the impact in clustering accuracy, regarding the choice of the similarity function used when

computing the similarity matrix. As mentioned earlier, we need a similarity function that is able to capture local edge structures in the network. For that, we tested three functions:

- Jaccard Similarity: the Jaccard Similarity coefficient of two nodes is the number of common neighbors divided by the number of nodes that are neighbors of at least one of the two nodes being considered;
- Dice's Coefficient (or Sørensen Index): the Dice Similarity coefficient of two nodes is twice the number of common neighbors divided by the sum of the degrees of the two nodes being considered;
- Inverse Log-weighted Similarity, as proposed in [1]: the inverse log-weighted similarity of two nodes is the number of their common neighbors, weighted by the inverse logarithm of their degrees. The intuition is that two vertices should be considered more similar if they share a low-degree common neighbor, since high-degree common neighbors are more likely to appear even by pure chance.

For each similarity function described above we ran Deep K-Means 10 times for the graphs g1 through g14 and measured the correspondent average accuracy as the normalized mutual information compared to the ground-truth communities.

For the auto-encoder pipeline of Deep K-Means we used a single-layer without random restarts.

3.3.3 Experiment #3

In this experiment we empirically analyze the time complexity of Deep K-Means compared to Spectral Clustering.

In Spectral Clustering we expect the time complexity being dominated by the following steps, which occur sequentially. Here we define n as the number of nodes in the graph and k, the number of communities:

- Construction of the similarity matrix: $O(n^2)$
- Eigenvalue decomposition of the similarity matrix: $O(n^3)$
- K-Means Clustering: $O(nk^2)$

For Deep K-Means, we have analogously:

- Construction of the similarity matrix: $O(n^2)$
- Deep K-Means pipeline: $O(rn^3)$, where r is proportional to the number of random initializations in the auto-encoder
- K-Means Clustering: $O(nk^2)$

We ran both Deep K-Means and Spectral Clustering 10 times for each network with the same setup as described in 3.3.1 and computed the average of the elapsed running time.

3.3.4 Experiment #4

In this experiment we empirically analyze the tradeoff between time complexity and accuracy, when running Deep K-Means with different configurations for the auto-encoder pipeline. More specifically, we vary both the number of layers and number of random initializations of K-Means for the auto-encoder.

We ran the first part of the test with the following parameters:

- Number of layers: 1, 2, 3, and 4
- Maximum number of iterations for K-Means on each layer: 25
- Number of random initializations for K-Means on each layer: 1
- Similarity function: Dice's

And the second part of the test with the following parameters:

- Number of layers: 1
- Maximum number of iterations for K-Means on each layer: 25
- Number of random initializations for K-Means on each layer: 1, 2, 4, 8, 32
- Similarity function: Dice's

For each part of the test described above, and for each corresponding parameter configuration, we ran Deep K-Means 10 times for the graphs g1 through g14 and measured the correspondent average accuracy and average execution time.

3.3.5 Experiment #5

In this experiment we analyze the performance of Deep K-Means and Spectral Clustering using the Football dataset [22].

The Football dataset is a representation of the schedule of Division I games for the 2000 season: vertices in the graph represent teams and edges represent regular season games between the two teams they connect. It incorporates a known community structure, which are "conferences" containing around 8 to 12 teams each. Games are more frequent between members of the same conference than between members of different conferences.

We ran Deep K-Means using a 1-layer auto-encoder and one K-Means initialization. We ran both Deep K-Means and Spectral Clustering 25 times and measure the NMI and execution time for each run.

3.4 Results

Here we present the results considerations for the experiments described above:

3.4.1 Result for Experiment #1

In Fig. 6 below we see that Deep K-Means outperforms Spectral Clustering in terms of clustering accuracy in a consistent way. We also notice that the accuracy of Spectral Clustering drops faster than the accuracy of Deep K-Means as the network increases in size, which makes Deep K-Means more robust when the network connectivity and community distribution complexity increase.

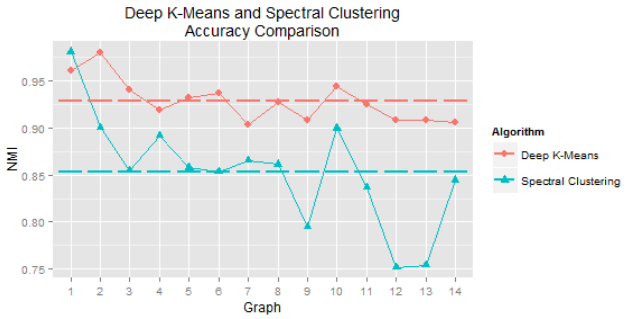

Fig. 6: *NMI accuracy for both algorithms; Graph represents the graph id as described in Tab.1; Each point in the plot represents the average NMI for 10 independent runs; dashed lines represent the accuracy average across all graphs for the corresponding algorithm*

3.4.2 Result for Experiment #2

In Fig. 7 below we see that the Dice's coefficient provides slightly better accuracy in most cases. This may indicate that giving more weight to the number of common neighbors between the nodes being compared, as Dice's similarity does when compared to Jaccard and Inverse Log-weighted, seems to be a better approach for Deep K-Means.

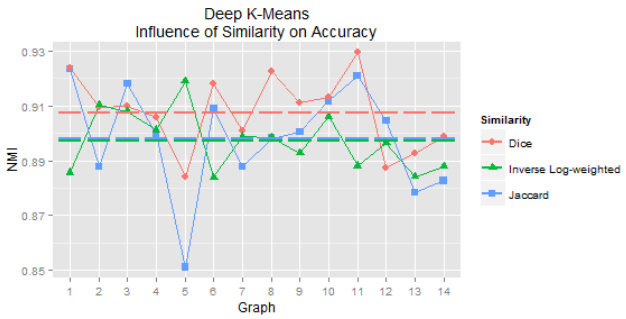

Fig. 7: *NMI accuracy for Deep K-Means using 3 different similarity scores: Dice's, Inverse Log-weighted, and Jaccard; dashed lines represent the accuracy average across all graphs for the corresponding similarity function*

3.4.3 Result for Experiment #3

In Fig. 8a and Fig. 8b below we notice the influence of the number of K-Means random initializations in the auto-encoder pipeline for Deep K-Means as the main difference in the execution time between both algorithms. As we notice in 3.4.4, there is no noticeable improvement in accuracy when increasing the number of random initializations, so that we could approximate the execution time of Deep K-Means much closer to Spectral Clustering by reducing the number of random initializations.

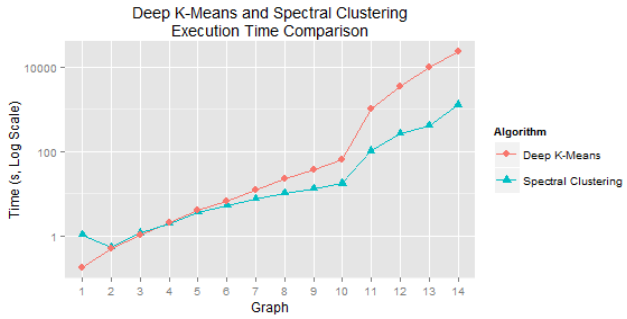

Fig. 8a: *Execution time for Deep K-Means and Spectral Clustering for graphs g1 trough g10*

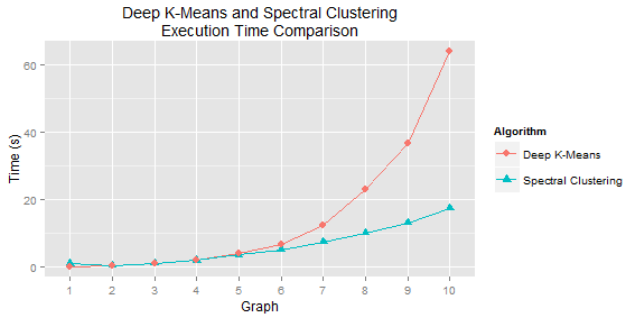

Fig. 8b: *Execution time for Deep K-Means and Spectral Clustering for graphs g1 trough g14 shown in logarithmic scale*

3.4.4 Result for Experiment #4

In Fig. 9a through Fig. 9f we analyze the behavior of Deep K-Means, when varying the number of layers and the number of random initializations in the auto-encoder pipeline, in terms of both accuracy and execution time.

We notice that the number of layers have very little influence on the accuracy, which is more noticeable when the number of nodes in the graph increases. Surprisingly, the configuration with 4 layers approaches the execution time of the single-layer configuration and even gets better, as the number of nodes in the network increases. This could be explained by the fact that increasing the number of auto-encoder layers eventually reduces the execution complexity of the last step of the algorithm (i.e, it converges faster), as defined in 3.2.g, to the point that the overall execution time can be reduced when compared to the execution time in a configuration with a smaller number of layers.

We also notice that the number of random initializations in the auto-encoder has no influence on the accuracy, but has a significant impact on execution time, which increases as a function of both the number of random initializations and the number of nodes.

Therefore we could improve the execution time of Deep K-Means in relation to Spectral Clustering without a significant change in accuracy, as shown in Fig. 9g and Fig. 9h.

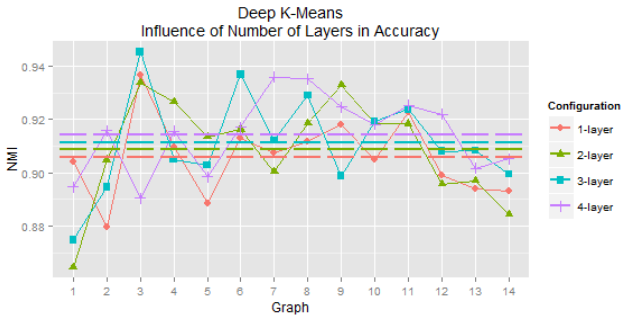

Fig. 9a: *NMI accuracy for Deep K-Means for 1, 2, 3, and 4 auto-encoder layers; dashed lines represent the accuracy average across all graphs for the corresponding configuration*

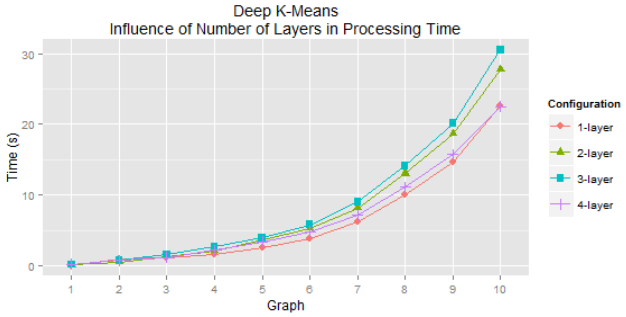

Fig. 9b: *Execution time for Deep K-Means for graphs g1 trough g10*

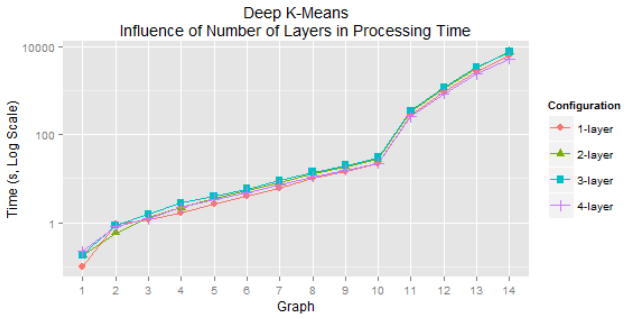

Fig. 9c: *Execution time for Deep K-Means for graphs g1 trough g14 shown in logarithmic scale*

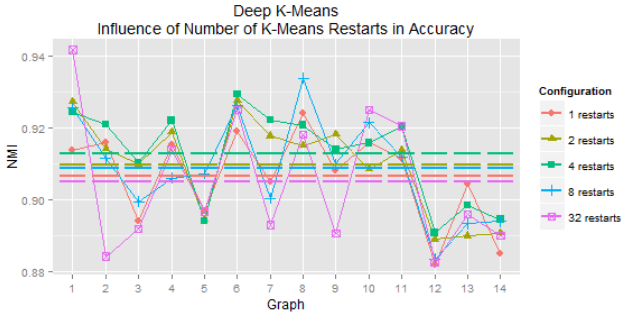

Fig. 9d: *NMI accuracy for Deep K-Means for 1, 2, 4, 8, and 32 random initializations; dashed lines represent the accuracy average across all graphs*

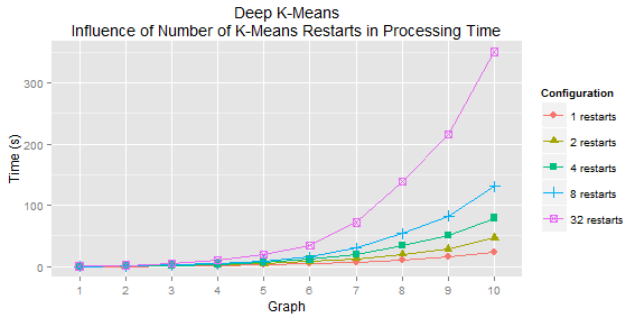

Fig. 9e: *Execution time for Deep K-Means for graphs g1 trough g10*

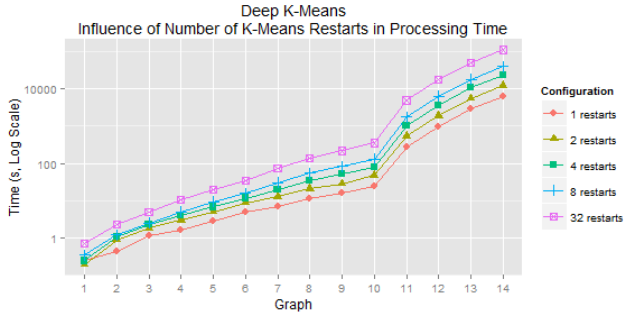

Fig. 9f: *Execution time for Deep K-Means for graphs g1 trough g14 shown in logarithmic scale*

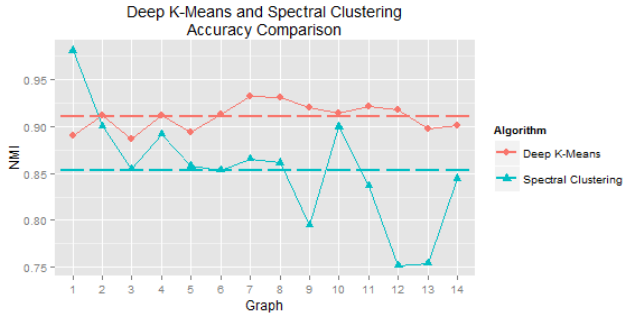

Fig. 9g *: NMI accuracy for Spectral Clustering and Deep K-Means with 4 layers and 1 random restart*

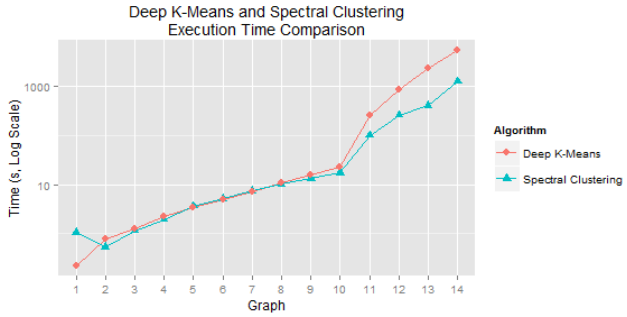

Fig. 9h: *Execution time for Deep K-Means and Spectral Clustering for graphs g1 trough g14 shown in logarithmic scale*

3.4.5 Result for Experiment #5

In Fig. 10a and Fig. 10b below we see the accuracy and execution time, respectively, when running both Deep K-Means and Spectral Clustering against the Football dataset [22]. As this is a fairly simple and small network, running Deep K-Means in a 1-layer configuration yields good results. As with the synthetic datasets in the previous experiments, we see Deep K-Means outperforming Spectral Clustering.

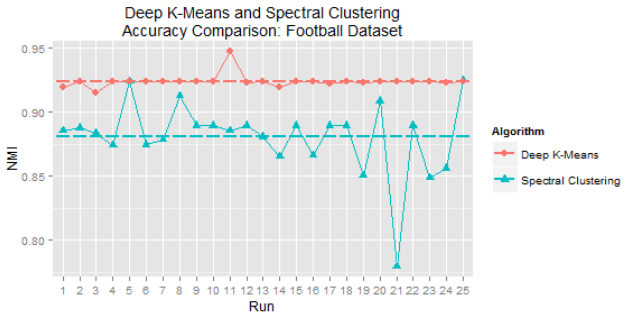

Fig. 10a: *community detection accuracy for Deep K-Means and Spectral Clustering for the 'Football' dataset; Each point in the plot represents the average NMI for 25 independent runs of Deep K-Means and Spectral Clustering; dashed lines represent the accuracy average across all runs*

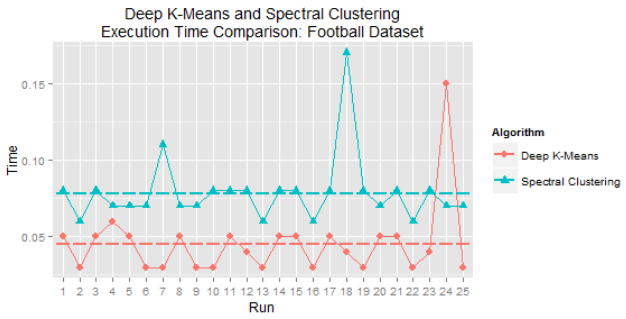

Fig. 10b: *Execution time for Deep K-Means and Spectral Clustering for the 'Football' dataset*

4. Conclusion

In this work we proposed a new algorithm for non-overlapping network community detection that leverages ideas from Deep Learning pipelines for data embedding in lower-dimensional spaces, which eases the task of clustering the data into communities.

We show through experiments using several datasets of varied sizes that the proposed algorithm outperforms traditional Spectral Clustering in accuracy measured by the Normalized Mutual Information between the communities discovered by the algorithm and the ground-truth communities associated with the respective datasets.

The drawback of the proposed approach is its execution time, which increases faster than Spectral Clustering as the datasets increase. Nevertheless, as the proposed approach is entirely based on K-Means clustering, it is much more easier to implement the algorithm in a distributed, parallel data processing framework, such as Apache Hadoop or Apache Spark, or to implement improved versions of K-Means such as proposed in [9].

The way the algorithm was implemented demands the prior knowledge of the number of communities to be found. This is not a problem for networks with ground truth communities, such as those tested in this work, but it is not the case for the majority of the practical problems. It would be interesting to incorporate some technique for automatic choosing the number of communities, such as optimizing the modularity function as proposed in [20] or the approach proposed in [10], where a statistical test is applied to each subset of the data assigned to a given cluster to assess if the number of clusters is optimal.

References

[1] Adamic, Lada A., and Eytan Adar. "Friends and neighbors on the web." *Social networks* 25.3 (2003): 211-230.

[2] Bengio, Yoshua. "Learning deep architectures for AI." *Foundations and trends® in Machine Learning* 2.1 (2009): 1-127.

[3] Boutsidis, Christos, et al. "Randomized Dimensionality Reduction for k-means Clustering." *arXiv preprint arXiv:1110.2897* (2011).

[4] Chen, Wen-Yen, et al. "Parallel spectral clustering in distributed systems."*Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33.3 (2011): 568-586.

[5] Coates, Adam, Andrew Y. Ng, and Honglak Lee. "An analysis of single-layer networks in unsupervised feature learning." *International Conference on Artificial Intelligence and Statistics*. 2011.

[6] Coates, Adam, and Andrew Y. Ng. "Learning feature representations with k-means." *Neural Networks: Tricks of the Trade*. Springer Berlin Heidelberg, 2012. 561-580.

[7] Erhan, Dumitru, et al. "Why does unsupervised pre-training help deep learning?." *The Journal of Machine Learning Research* 11 (2010): 625-660.

[8] Fortunato, Santo. "Community detection in graphs." *Physics Reports* 486.3 (2010): 75-174.

[9] Frahling, Gereon, and Christian Sohler. "A fast k-means implementation using coresets." *International Journal of Computational Geometry & Applications*18.06 (2008): 605-625.

[10] Hamerly, Greg, and Charles Elkan. "Learning the k in A> means." *Advances in neural information processing systems* 16 (2004): 281.

[11] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507.

[12] Jain, Anil K. "Data clustering: 50 years beyond K-means." *Pattern Recognition Letters* 31.8 (2010): 651-666.

[13] Lancichinetti, Andrea, Santo Fortunato, and Filippo Radicchi. "Benchmark graphs for testing community detection algorithms." *Physical Review E* 78.4 (2008): 046110.

[14] Ng, Andrew Y., Michael I. Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm." *Advances in neural information processing systems* 2 (2002): 849-856.

[15] Song, Yangqiu, et al. "Parallel spectral clustering." *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2008. 374-389.

[16] Tian, Fei, et al. "Learning Deep Representations for Graph Clustering." *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

[17] Tsironis, Serafeim, Mauro Sozio, and Michalis Vazirgiannis. "Accurate Spectral Clustering for Community Detection in MapReduce."

[18] Von Luxburg, Ulrike. "A tutorial on spectral clustering." *Statistics and computing* 17.4 (2007): 395-416.

[19] Weston, Jason, et al. "Deep learning via semi-supervised embedding." *Neural Networks: Tricks of the Trade*. Springer Berlin Heidelberg, 2012. 639-655.

[20] White, Scott, and Padhraic Smyth. "A Spectral Clustering Approach To Finding Communities in Graph." *SDM*. Vol. 5. 2005.

[21] Yang, Jaewon, and Jure Leskovec. "Defining and evaluating network communities based on ground-truth." *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*. ACM, 2012.

[22] Girvan, Michelle, and Mark EJ Newman. "Community structure in social and biological networks." Proceedings of the National Academy of Sciences 99.12 (2002): 7821-7826.