NEURALLY BOOSTED SUPERVISED SPECTRAL CLUSTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Network embedding methods compute geometric representations of graphs that render various prediction problems amenable to machine learning techniques. Spectral network embeddings are based on the computation of eigenvectors of a normalized graph Laplacian. When coupled with standard classifiers, spectral embeddings yield strong baseline performance in node classification tasks. Remarkably, it has been recently shown that these 'base' classifications followed by a simple 'Correction and Smooth' procedure reach state-of-the-art performance on widely used benchmarks. All these recent works employ classifiers that are agnostic to the nature of the underlying embedding. We present simple neural models that leverage fundamental geometric properties of spectral embeddings and obtains significantly improved classification accuracy over commonly used standard classifiers. Our results are based on a specific variant of spectral clustering that is not well-known, but it is presently the only variant known to have analyzable theoretical properties. We provide a `PyTorch` implementation of our classifier along with code for the fast computation of spectral embeddings.

1 INTRODUCTION

Spectral clustering is a classical unsupervised algorithm for clustering in graphs. It gained prominence in data mining and machine learning approximately two decades ago (Shi & Malik, 2000; Ng et al.) and it has since found many applications in multiple domains. While originally conceived as an unsupervised method, spectral clustering has also been employed in transductive learning, and in particular in semi-supervised node classification that is the focus of this work.

Background. The work of Tang & Liu (2011) was likely what initiated a long line of work on multi-label classification with experimental focus on social network benchmarks. They described a general framework for node classification consisting of two steps: (i) computing a network embedding, (ii) using the embedding as input to a standard classifier. They instantiated step (i) with a spectral embedding, giving thus rise to what can be viewed as a *supervised spectral clustering* algorithm. Albeit simple, the algorithm proved to be superior relative to the then-available baselines. The thread continued with the very prominent DEEPWALK embedding algorithm (Perozzi et al., 2014), that claimed superiority over spectral clustering which however remained competitive. The success of DEEPWALK unleashed a wave of research works on network embeddings, including (Grover & Leskovec, 2016; Tang et al., 2015; Tsitsulin et al., 2018; Qiu et al., 2018a), and a race for higher classification accuracy [1]. More recently, network embedding methods were succeeded by graph neural networks (GNNs) (e.g. see (Kipf & Welling, 2017; Velickovic et al., 2018)) that have dominated the competition. Since the publication of DEEPWALK, supervised spectral clustering was at-best considered a baseline, or entirely vanished from consideration. However it has recently made a spectacular comeback with the work of Huang et al. (Huang et al., 2021) that described a simple 'correction and smooth' (C&S)

[1] A very detailed list can be found in `https://github.com/chihming/awesome-network-embedding`

procedure that can enhance the output of other models. Some of the state-of-the-art results in in the Open Graph Benchmark leaderboard (Hu et al., 2021) are now based on variants of supervised spectral clustering in combination with the C&S process. These easily surpass previous embedding methods and even exceed methods based on GNNs using only a small fraction of their number of trainable parameters.

A Critical Look. This work originates from the observation that previous literature has treated spectral clustering somewhat casually, and largely ignoring recent major advances in its theoretical understanding. As discussed above, supervised spectral clustering consists of two steps: (i) Computing a spectral embedding, and (ii) Giving the embedding as input to a multi-label classifier. In the literature, the term 'spectral clustering' is used for multiple different algorithms that implement steps (i) and (ii) in various and sometimes not adequately described ways.

For example, for step (i), (Tang & Liu, 2011) and (Perozzi et al., 2014) use eigenvectors of the normalized graph Laplacian as proposed in (Ng et al.) but appear to omit a normalization step from that algorithm, (Huang et al., 2021) also use the normalized graph Laplacian but with an additional 'rotation' transformation, while other works use the random walk matrix (i.e. the non-symmetric normalization of the Laplacian) which in fact is a more accurate relaxation of the underlying NCut problem (Shi & Malik, 2000). Classifiers used for step (ii) include an One-vs-All SVM-based model (Tang & Liu, 2011), an One-vs-Rest logistic regression model (Perozzi et al., 2014), a softmax regression model, and a three-layered neural network (Huang et al., 2021). Adding to the confusion, the computation of eigenvectors has been viewed both as computationally expensive to the point of preventing its application on larger networks (Perozzi et al., 2014), or as able to scale up to very large networks via standard iterative eigensolvers whose fast convergence cannot however be guaranteed (Huang et al., 2021).

Goal and Contribution. These considerations provide motivation of our work. Concretely, we seek to provide a theory-informed implementation of supervised spectral clustering, maximize its performance with the use of a carefully designed neural model and study its practical performance in the context of multi-label classification on social networks.

Conceptually our work can be viewed as a 'neural' implementation, in the supervised setting, of the seminal theoretical work of Lee et al. (2014) on multi-way Cheeger inequalities for spectral clustering. These theoretical guarantees on the performance of spectral clustering are based on a variant of spectral clustering which is not well-known or widely used, yet it is the only variant that can be analyzed in the general case, i.e. without assumptions about the underlying graph. The analysis of Lee et al. applies to unsupervised clustering. Thus the main contribution of our work is to transfer the geometric intuition developed in (Lee et al., 2014) to the supervised setting with the design of custom-made neural classifiers that respect and exploit the geometric facts.

Indeed, we show experimentally that the proposed combination of spectral embedding and neural classifiers increases significantly the performance of supervised spectral clustering to previously unobserved levels. And while our goal is not to claim any type of 'supremacy' in the accuracy race, we do show that –within our experimental context– supervised spectral clustering yields higher accuracy than several prominent alternative embedding-based methods that were thought to be superior. We thus hope that our work can positively influence future research in this very active area.

2 Spectral Clustering with Supervision: The algorithm

2.1 Spectral Embedding and Geometric Intuition

We begin with a review of the spectral embedding algorithm. Along it we give concrete details on how it can be implemented to run in near-linear time with strong convergence guarantees. In Section 2.1.3 we discuss the geometric intuition that underlies the theoretical analysis in (Lee et al., 2014). We note that the

material in this subsection is not new. We present it for completeness and in particular in order to discuss the geometric intuition which plays a key role in the construction of our classifiers.

2.1.1 DEFINITIONS AND TERMINOLOGY

Let $G = (V, E, w)$ be a weighted undirected graph.

Definition 1 (**Cut and Volume**). *If $S \subset V$, we define the cut of S to be*

$$cut(S) = \sum_{i \in S, j \notin S} w_{i,j}$$

We also define the volume of S to be

$$vol(S) = \sum_{s \in S} cut(v).$$

*We will often use the term **degree** for the cut of a single node.*

Definition 2 (**Conductance**). *The conductance of a cluster $S \subset V$ is defined by*

$$\phi(S) = \frac{cap(S, V - S)}{\min(vol(S), vol(V - S))}$$

We denote by D the diagonal matrix of the degrees, by d the vector of the node degrees, and by $L = D - A$ the graph Laplacian, where A is the adjacency matrix of the graph.

2.1.2 SPECTRAL EMBEDDING AND THE UNSUPERVISED ALGORITHM

The combinatorial idea that underlies spectral clustering is that good clusters should have a relatively small volume to their exterior, or more specifically a low conductance. More concretely, the goal is to find k disjoint clusters $\{S_1, S_2, ...S_k\}$ such that $\max_i \phi(S_i)$ is as small as possible. In this paper we will be looking for k clusters that also provide a partitioning of the vertex set, i.e. we have $\bigcup_i S_i = V$. The partitioning problem is less well-understood (Louis et al.), but its simpler non-partitioning variant has been studied extensively from a theoretical perspective (Lee et al., 2014), and approximation guarantees are known for it. The spectral clustering algorithm is broadly understood as a relaxation of the discrete clustering problem (e.g. see (Cucuringu et al.) for a complete discussion), and a number of variants have been proposed (von Luxburg, 2007). In its simplest form, the variant that is analyzed in (Lee et al., 2014) consists of three steps, summarized here:

- *(a) Spectral Embedding.* Compute the eigenvectors x_j corresponding to the k smallest non-zero eigenvalues of the generalized problem $Lx = \lambda Dx$, under the constraint that $x^T d = 0$. The eigenvectors are normalized so that $x_j^T D x_j = 1$. Let $X \in \mathbb{R}^{n \times k}$ be the matrix whose columns are the k eigenvectors.
- *(b) Orthogonal Projection.* Let $X' = XR$, where $R' \in \mathbb{R}^{k \times k'}$a random orthogonal projection matrix, where $k' \leq k$.
- *(c) Radial Projection.* Let Y be the matrix resulting by dividing each row of X by its Euclidean norm, i.e. by 'radially projecting' each point onto the unit sphere in $\mathbb{R}^k$. Each row of Y becomes the embedding of the corresponding node.
- *(d) Geometric Partitioning.* Use some unsupervised geometric clustering algorithm on the embedding Y. (e.g. k-means, or the provable algorithms presented in (Lee et al., 2014)).

We remark that the two hyperparameters k and k' are further specified in sections 2.2 and 3.1.

2.1.3 THE CRUCIAL GEOMETRIC INSIGHT

The radial projection step reflects the geometric intuition that the eigenvectors place the nodes into k *directions* in the embedding space.

Consider the eigenvector matrix $X \in \mathbb{R}^{n \times k}$. These generalized eigenvectors are D-orthogonal (Stewart & Sun, 1990), i.e. we have $x_i^T D x_j = 0$, for each $i \neq j$. Given that the eigenvectors are also normalized, we have $X^T D X = I$.

Using a well known identity we have

$$trace(X^T D X) = trace(D X X^T) = k.$$

Summing up the n diagonal entries of $D X X^T$ we thus get

$$\sum_{j=1}^{n} d_j ||x_j||_2^2 = k. \tag{1}$$

Now pick an arbitrary direction in $\mathbb{R}^k$, i.e. a unit vector z. Let $w = Xz$. Since $X^T D X = I$, and $z^T z = 1$ we have

$$z^T X^T D X z = w^T D w = \sum_{j=1}^{n} d_j w_j^2 = 1. \tag{2}$$

Let us now take a closer look at the above equations. From equation 1 we get that that the d-weighted mass of the n embedding points is k. On the other hand, note that w_j^2 is the norm of the projection of $X[j]$ onto z. Thus, equation 2 shows that the d-weighted mass of the projections of the n points onto z is always equal to 1, for any direction z.

A further interpretation of the above equations is that any given direction z captures only a $1/k$ fraction of the d-weighted sum of norms of the n embedding points. Hence the d-mass of the points must concentrate in at least k different directions. While finding such directions is not straightforward, (Lee et al., 2014) proves that radially projecting the points onto the unit sphere has the following property: if S is a set of points in Y that are in a ball of a small diameter in $\mathbb{R}^k$, then S also cannot concentrate a large fraction of the d-mass of the points, and thus the points in Y are expected to be in k well-separated spheres in $\mathbb{R}^k$. This specific geometric arrangement facilitates the design of an unsupervised geometric clustering algorithm on the points of Y in (Lee et al., 2014), but it will also guide the design of our supervised geometric clustering model.

2.1.4 EFFICIENT COMPUTATION

We briefly discuss the computation of eigenvectors of the generalized eigenvalue problem $Lx = \lambda Dx$. Several works compute eigenvectors using implementations of ARPACK (Lehoucq et al., 1997) in libraries for sparse matrix algebra, in Matlab, Python, or Julia (e.g. (Huang et al., 2021)). Social networks usually have eigenvalue distributions that are 'friendly' to power methods (that are used in ARPACK) in the sense that they can converge in a reasonable amount of time. However that is not guaranteed, and in practice convergence can be much slower than possible. Fast convergence can presently be guaranteed only via using inverse-power algorithms, i.e. algorithms that use approximations of the inverse of the Laplacian, often used as preconditioners. In this work we use the preconditioned eigensolver LOBPCG (Knyazev, 2001) with the CMG preconditioner (Koutis et al., 2011). Besides its stronger theoretical properties, the particular implementation is also significantly faster in practice, even on social networks.

2.2 NEURAL CLASSIFIERS

We present three types of neural classifiers that reflect our discussion in section 2.1. These are different but they share certain layers and so they are presented in a compact way in Algorithm 1 that gives pseudocode for their 'forward' operation. Below we make our main points on the design of these classifiers.

- Step (ii) in the unsupervised algorithm described in section 2.1.2 is a random orthogonal projection. In analogy, we introduce a linear operator R in line 11 of Algorithm 1 which in our case is entirely trainable. We mark line 11 as optional, but our ablation study (see section 3.3) shows that it is essential for higher accuracy.

- The radial projection, step (iii) in in section 2.1.2, is implemented in line 12. In our ablation study we also run experiments without this step and they confirm that radial projection is key to higher accuracy.

- The above two steps are reproduced independently in K independent channels, where K is a hyperparameter. Each of these channels has its own operator R_i. In our ablation study we find that adding channels in general increases the accuracy of the classifier.

- The code presents three different types of classifiers: *Conic*, *Linear*, and *Spherical*. These share the same parameters W and β. The outer dimension of W is proportional to the number of channels, but the inner dimension of both W and β are equal to the number of classes. What differentiates these three types of classifiers is the role of W and β. Their names reflect the corresponding geometric shape of the decision boundaries.

- *Linear:* Each of the nc rows of W defines a hyperplane in $\mathbb{R}^{K \times nc}$ as its normal vector. The decision boundary for row W_j is the hyperplane $yW + \beta$.

- *Conic:* Row W_j represents represents a direction in $\mathbb{R}^{K \times nc}$. The decision boundary is given by equation $yW_j - ||W_j||_2 cos(\beta)$, which is well-understood to describe the surface of a cone with axis W_j and aperture β, given that y has unit norm (line 12). We note that the decision boundary can be rewritten as $yW_j + \beta'$, where $\beta' = ||W_j||_2 cos(\beta)$. Thus it can be seen that *Conic* is a restricted or 'regularized' version of *Conic* because it enforces $\beta' \leq ||W_j||_2$.

- *Spherical:* Row W_j represents a point in $\mathbb{R}^{K \times nc}$, which is the center of the sphere with of radius β. The decision boundary is the sphere $\beta^2 - ||W_j - y||_2^2$.

2.3 INITIALIZATION

The discussion in previous sections allows us certain considerations on the initialization of the trainable parameters. In the unsupervised algorithm, the matrix R (in line 11) is taken to be a random orthogonal projection, and we can thus initialize it as such. Moreover, according to the discussion in section 2.1.3, the node embeddings given by XR are expected to concentrate in at least nc different directions/ geometric clusters after the radial projection step. Approximate directions or 'centers' for these clusters can be found using the embeddings of the training points. Specifically, in order to compute such approximate centers or directions, we take the radial projection of the training points for each cluster separately, compute their average, and then we radially project it. These initializations can be carried out independently for each of the K channels. In our experiments we have used such initializations; they do not affect performance, but seem to improve convergence speed. More details will be discussed in the full version of the paper.

Algorithm 1 Radial-Conic-Spherical Classifier with K channels

Input: $x \in \mathbb{R}^{1\times k}$: eigenvector embedding, $type$: 'Linear' or 'Conic'
Output: $y \in \mathbb{R}^{1\times nc}$: nc is the number of classes

Trainable parameters:
$R_i \in \mathbb{R}^{k\times nc}$, for $i = 1, \ldots K$, where K is a hyperparameter
$W \in \mathbb{R}^{K\cdot nc\times nc}$
$\beta \in \mathbb{R}^{1\times nc}$

function RCS-CLASSIFY(x)
  for i=1 **do** to K
    $y = xR_i$ ▷ [optional]
    $y_i = y/||y||_2$ ▷ radial projection
  end for
  $y = [y_1; \ldots; y_K]$ ▷ concatenate
  if type = 'Conic' **then**
    $w_{nrm} = norm(W, dim = 1)$ ▷ $w_{nrm} \in \mathbb{R}^{1\times nc}$: vector of 2-norms of columns of W
    $y = yW - w_{nrm}. * cos(\beta)$ ▷ $.*$ denotes pointwise multiplication
  else if type = 'Linear' **then**
    $y = yW + \beta$
  else if type = 'Spherical' **then**
    $y = \beta^2 - (W - \mathbf{1}^{K\cdot nc\times 1}y)^2$ ▷ squaring is entry-wise
  end if
  $y = \mathbf{softmax}(y)$
end function

3 EXPERIMENTS

The experiments reflect the main message of this work that the theoretical understanding of the geometric properties of spectral clustering can render it much more powerful on social graphs, relative to what has been observed earlier in the literature.

We specifically want to maximize classification performance **on the graphic data only**, excluding additional node features that are often provided for social graphs. In other experiments we have found that extending our method to include features as an additional 'channel' in RCS-CLASSIFY (algorithm 1) can indeed enhance accuracy performance; however such inclusion requires systematic ideas that are beyond the scope of this paper and will likely obscure our main message; for that reason we do not include such node features in our experiments.

3.1 EXPERIMENTAL SETUP

Datasets and Hyperparameters. We use well-known benchmarks and follow standard practices. We refer the reader to the Appendix A for detailed information.

Evaluation Method. We measure the accuracy of the various methods trained with randomly selected training sets at label rates: $2^j\%$, for $j = 0, \ldots, 5$. For each label rate, we pick 5 different random training sets. For each of these training sets we train each classifier for 5 different random initializations of the trainable weights. For each such experiment, we perform 3-fold cross validation where each validation set is selected randomly and consists of 20% of the training set (i.e. 20% of the label rate). We ensure that

every class is represented in our training set by forcing one random node from each class in the training set. Validation is used for RCS-CLASSIFY in order to select the best-performing model over the course of training.

3.2 METHODS

This subsection describes the different methods appearing in Table 1.

Rows 1-9: LINEAR NRP runs the linear classifier in algorithm 1, excluding lines 10-13; that means that there is no reduction in the dimension and no radial projection (NRP). LINEAR runs the linear classifier in algorithm 1, excluding line 11. CONIC and SPHERICAL are the corresponding branches of algorithm 1; if more than one channel is used, that is noted in Table 1.

Rows 10-11: These denote a simple logistic regression classifier, with the one-vs-rest method used for multilabel classification. Specifically we use the well known LIBLINEAR library. These do not use any validation set. In both cases we used the radially projected spectral embedding as input. The difference between row 10 and 11 is the use (or not) of regularization during training.

Rows 12-13: The standard kNN-classifier with $k = 11$, that we have found to be a universally good setting of the hyperaparameter. The input embedding is the radially-projected spectral embedding.

Rows 2,4,6,9,13: Adding the C&S procedure (Huang et al.) on top of the preceding classifier.

Rows 14-16: Here we present other methods, not based on the spectral embedding. Specifically we consider these algorithms: DeepWalk (Perozzi et al., 2014), NetMF (Qiu et al., 2018b), a standard GCN model (Kipf & Welling, 2017) and the RANDOMWALKER algorithm (harmonic label interpolation) (Grady, 2006). We note that, as discussed in (Qiu et al., 2018b), both DeepWalk and NetMF essentially compute embeddings of a log-transformed power of the random walk matrix, and for that reason they can be in general more powerful, but also computationally expensive. We note that the C&S procedure cannot be applied on top of the LIBLINEAR classifier, but it can be applied on top of the GCN, and possibly on top of the NetMF method, assuming using a different classifier than LIBLINEAR that was used in the original paper. However these experiments are not undertaken here.

3.3 RESULTS

We experimented with 16 different classification methods, 6 different datasets and 6 different label rates. These experiments are presented in Table 1. We point the reader to section 3.2 for details on the different methods used. We note that we omit information on the variance over random splits and random initializations. This is in general very low (< 0.5) with the exception of the CORA and CITESEER datasets that we discuss separately. We also note that some of the experiments on the PRODUCTS datasets have not been completed due to limited resources (see Appendix A; the final paper will contain all these missing items, with the exception of rows 16-17, which is generally impossible due to the complexity of NETMF and DEEPWALK.

A close look at the table leads to several interesting observations, that we summarize here:

• Rows 1,3,5,7,8 can regarded as an ablation study on the methods encapsulated in RCS-CLASSIFY (algorithm 1). In general, we expect that the inclusion of the radial projection step, the reduction layer R_i, and more channels (as grey-shading becomes darker), should improve performance. At the same time, we should be aware of the fact that these additional parameters entail the risk of overfitting on smaller training sets.

Radial projection clearly appears to help universally, with the exception of CORA. The additional parameters of CONIC do seem to create an overfitting issue in the smaller datasets CITESEER and CORA. In particular, CORA is an extreme outlier; we believe that the low observed accuracy is due to overfitting, because the

KNN and LIBLINEAR classifiers (applied on the same embedding) appear to do much better. However, these additional features clearly help on the bigger training sets, with the 2-channel CONIC (line 8) showing a very clear dominance for the largest training sets of the ARXIV dataset, and a very big improvement with respect to standard classifiers (e.g. 62.6% at 4% label rate vs 47.3% for the commonly used LIBLINEAR).

• Rows 10-11 consider LIBLINEAR, which in the literature has been employed with the use of regularization, as in row 10. We see that with the exception of CITESEER and CORA, the use of regularization is detrimental to the accuracy of the method, a fact that provides further evidence that these smaller datasets induce overfitting issues, and corroborates the theoretical insight on the value of the radial projection step.

• CONIC's accuracy is only slightly better relative to the accuracy of the full LINEAR classifier in algorithm 1 (i.e. without dropping any parts of the algorithm as in rows 1-4). That is to be expected, given their similarity (section 2.2). On the other hand, SPHERICAL which is mathematically quite different from CONIC reaches a higher accuracy relative to CONIC, in most cases. That is remarkable considering that CONIC and SPHERICAL have identical numbers of parameters. We note however that we have trained the SPHERICAL model for 4K epochs, unlike other models that have trained for 2K epochs. That is because, we have empirically found that SPHERICAL shows a slower convergence. In that sense, the dominance of SPHERICAL is not unequivocally demonstrated, and for that reason we did not pursue 2 or more channels with SPHERICAL. However, SPHERICAL is indeed a new kind of classifier that we believe merits a further study.

• With the exception of CORA, the standalone (non-C&S) SPHERICAL or CONIC is in most cases better that the other models, including GCNs. GCNs have of course dominated the state of the art with their performance when node features are taken into consideration, but the performance of our spectral-based neural models (even without the C&S process) reaches previously unseen and possibly surprising levels.

• Further unreported experiments show that adding more than 2 channels in our neural models can further increase accuracy on the bigger datasets, while increasing in a natural and controlled way the number of parameters. For instance, at the 32% label rate for ARXIV, we have observed that accuracy keeps rising by adding up to 10 channels.

• The C&S procedure is in general very effective, as also observed in (Huang et al., 2021). The final test performance of each model after the application of C&S procedure seems to be generally correlated with the 'base' model performance, without the C&S procedure. That motivates further research on standalone no-C&S classifiers It is also remarkable that the enhancement provided by the C&S procedure appears to vanish as the base models become stronger as also observed in (Huang et al., 2021).

• Some additional remarks are discussed in Appendix B.

4 CONCLUSION

Research on transductive node classification tasks has recently been dominated by Graph Neural Networks and the evolution of progressively more elaborate and highly-parameterized neural architectures that yield end-to-end trainable network embeddings. This is in stark contrast to the pre-GNN period, when the collective focus was on the design of network embedding methods, with apparently little research devoted to the downstream classification task. In this work we argued that developing a better theoretical understanding of the geometric embedding opens up the fundamental research direction of designing better geometry-aware classifiers. This direction can lead to surprising substantial improvements, even for older algorithms, such as spectral clustering. In principle, this reasoning can be applied to other existing network embedding methods with the potential to –more broadly– inform and steer the design of improved graph neural architectures.

Classifier	CITESEER						CORA						WIKICS					
	1%	2%	4%	8%	16%	32%	1%	2%	4%	8%	16%	32%	1%	2%	4%	8%	16%	32%
1. LINEAR NRP	56.4	50.3	51.0	54.3	57.4	58.4	16.7	23.9	25.7	28.6	26.8	29.0	29.7	34.3	35.0	37.7	35.8	35.9
2. LINEAR NRP +CS	58.4	64.5	**69.3**	**70.4**	**72.9**	**74.2**	69.5	74.2	75.6	78.3	81.4	83.2	52.1	52.2	54.8	57.3	60.0	71.5
3. LINEAR	49.0	63.6	66.9	68.9	68.9	71.1	18.9	18.8	18.8	17.5	25.5	27.4	69.4	71.7	73.9	74.8	75.8	76.2
4. LINEAR +CS	57.6	64.9	68.7	70.6	72.2	74.1	69.3	73.6	75.2	77.9	**82.7**	**84.8**	69.2	73.6	76.9	77.6	80.0	81.8
5. CONIC	45.8	59.6	64.6	67.6	69.7	70.7	16.0	19.0	19.0	22.7	24.4	26.7	69.9	73.2	75.0	75.4	78.2	78.7
6. CONIC +CS	55.9	64.3	67.7	70.2	72.1	74.0	64.9	72.2	75.1	80.1	82.7	84.7	71.1	**74.3**	**77.8**	**78.6**	**80.5**	81.9
7. SPHERICAL	45.0	62.3	65.1	67.9	69.0	70.9	18.1	19.6	22.3	24.5	26.3	28.1	68.8	73.2	75.3	76.9	78.5	79.0
8. CONIC 2 CHANNEL	48.1	60.9	65.3	68.2	69.9	71.1	17.1	17.9	20.2	21.7	21.7	27.0	70.5	74.0	75.0	76.1	78.4	79.8
9. CONIC 2 CHANNEL +CS	55.9	63.5	67.2	69.2	71.3	73.7	56.9	42.2	73.3	79.2	82.0	84.8	70.3	74.0	76.9	77.6	80.3	**82.3**
10. LIBLINEAR (REG)	55.3	64.4	67.8	68.7	69.8	70.9	63.6	68.8	71.5	74.1	75.0	76.0	**72.8**	73.6	74.9	75.4	76.1	76.3
11. LIBLINEAR (NO-REG)	54.5	56.7	65.0	68.6	70.4	71.1	64.4	65.1	71.0	75.7	78.2	78.3	67.5	71.0	75.0	76.2	76.9	77.0
12. KNN (K=11)	49.6	62.2	67.3	68.7	69.8	70.7	50.9	66.9	70.7	74.8	77.3	78.9	70.3	72.7	74.6	76.6	78.5	79.5
13. KNN (K=11+CS)	57.8	62.9	65.9	67.8	69.5	72.1	**71.7**	**75.2**	**76.3**	**79.9**	82.4	84.4	73.2	74.6	76.4	78.7	80.4	81.4
14. RANDOM WALKER	**60.7**	**65.0**	67.1	70.0	71.3	73.0	68.5	72.8	77.4	79.6	81.9	83.1	21.3	32.2	42.6	63.8	73.7	79.1
15. GCN	41.3	44.5	55.9	60.5	65.8	68.9	15.3	16.3	16.6	16.3	16.0	16.8	65.1	69.1	72.7	76.4	78.0	79.0
16. NETMF	39.4	49.8	53.3	54.5	57.1	60.8	61.2	66.1	69.5	71.6	74.8	76.8	70.3	68.5	68.6	69.5	73.6	80.7
17. DEEPWALK	49.3	47.1	54.1	63.1	65.8	67.4	61.9	69.2	73.7	77.4	78.5	79.0	72.1	75.0	77.0	78.2	79.2	79.8

Classifier	PUBMED						ARXIV						PRODUCTS					
	1%	2%	4%	8%	16%	32%	1%	2%	4%	8%	16%	32%	1%	2%	4%	8%	16%	32%
1. LINEAR NRP	53.2	53.6	59.0	55.2	58.8	58.5	21.2	26.5	26.0	24.6	25.8	25.1	27.9	28.7	31.1	28.9	-	-
2. LINEAR NRP + CS	76.0	77.9	**79.1**	80.0	80.7	81.5	53.2	59.8	63.7	66.0	68.4	70.4	83.7	83.6	83.8	85.0	-	-
3. LINEAR	63.5	64.6	63.4	64.3	65.2	64.0	47.6	48.5	49.0	49.3	49.3	49.4	66.6	66.7	66.6	66.7	-	-
4. LINEAR + CS	76.0	77.7	79.4	**80.5**	**81.4**	**82.3**	56.4	61.0	64.8	67.4	69.6	71.3	84.8	85.7	86.5	87.4	-	-
5. CONIC	75.6	75.4	76.0	76.0	76.6	76.6	58.4	60.5	61.3	62.1	62.4	62.5	80.9	80.9	80.9	81.0	-	-
6. CONIC + CS	**76.9**	**77.6**	78.8	80.1	81.0	82.3	61.0	63.3	65.4	67.0	69.5	71.3	**85.6**	**86.2**	**86.8**	**87.2**	-	-
7. SPHERICAL	75.0	75.0	75.5	76.0	76.1	76.3	57.8	60.5	61.9	63.0	63.7	63.8	-	-	-	-	-	-
8. CONIC 2 CHANNELS	75.2	75.3	76.0	76.4	76.8	76.8	58.6	60.7	62.6	64.1	64.9	65.4	-	-	-	-	-	-
9. CONIC 2 CHANNEL + CS	76.6	77.4	78.3	79.9	81.0	82.3	**61.1**	**63.3**	**65.6**	**67.6**	**69.6**	**71.3**	-	-	-	-	-	-
10. LIBLINEAR (REG)	60.3	60.6	61.7	64.3	67.1	68.8	43.8	45.4	47.3	48.9	50.2	51.4	67.1	69.4	73.4	75.4	76.5	77.3
11. LIBLINEAR (NO-REG)	73.3	73.4	72.7	72.1	71.6	71.5	56.9	58.7	59.2	59.1	59.2	59.2	79.9	79.9	80.0	80.1	80.1	80.1
12. KNN (K=11)	73.5	74.7	75.4	76.3	77.3	78.3	54.7	57.3	59.2	60.9	62.6	64.3	80.7	81.9	83.0	84.1	85.0	85.9
13. KNN (K=11+ CS)	76.0	76.6	77.9	79.4	80.7	81.9	59.5	62.4	64.8	66.8	68.7	70.6	-	-	-	-	-	-
14. RANDOM WALKER	66.8	73.0	77.0	79.4	80.8	81.8	32.3	41.3	52.2	60.9	66.0	69.3	79.1	83.8	85.3	86.7	**87.9**	**89.0**
15. GCN	69.0	72.8	75.3	77.3	79.1	80.3	56.8	60.1	62.1	63.6	64.3	64.9	-	-	-	-	-	-
16. NETMF	75.7	76.1	76.5	76.6	76.7	76.7	-	-	-	-	-	-	-	-	-	-	-	-
17. DEEPWALK	77.2	77.3	77.7	77.8	77.9	78.0	-	-	-	-	-	-	-	-	-	-	-	-

Table 1: Best performance is marked with bold.
Beyond that, a further interesting picture emerges - and it is discussed in section 3.3

REFERENCES

Mihai Cucuringu, Ioannis Koutis, Sanjay Chawla, Gary Miller, and Richard Peng. Simple and Scalable Constrained Clustering: a Generalized Spectral Method. 51(Proceedings of the 19th International Conference on Artificial Intelligence and Statistics):445–454. URL http://proceedings.mlr.press/v51/cucuringu16.html.

Leo Grady. Random Walks for Image Segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2(11):1768–1783, 2006.

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*, KDD '16, pp. 855–864, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939754. URL http://doi.acm.org/10.1145/2939672.2939754.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv:2005.00687 [cs, stat]*, February 2021. URL http://arxiv.org/abs/2005.00687. arXiv: 2005.00687.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin Benson. Combining Label Propagation and Simple Models out-performs Graph Neural Networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=8E1-f3VhX1o.

Xiao Huang, Jundong Li, and Xia Hu. Label Informed Attributed Network Embedding. doi: 10.1145/3018661.3018667. URL http://dx.doi.org/10.1145/3018661.3018667.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Andrew V Knyazev. Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method. *SIAM Journal on Scientific Computing*, 23(2):517–541, 2001. doi: 10.1137/S1064827500366124. URL https://doi.org/10.1137/S1064827500366124.

Ioannis Koutis, Gary L. Miller, and David Tolliver. Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. *Computer Vision and Image Understanding*, 115(12):1638–1646, 2011.

James R. Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway Spectral Partitioning and Higher-Order Cheeger Inequalities. *ACM*, 61:37:1–37:30, 2014.

R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*. Available from netlib@ornl.gov, 1997.

Anand Louis, Prasad Raghavendra, Prasad Tetali, and Santosh Vempala. Algorithmic Extensions of Cheeger's Inequality to Higher Eigenvalues and Partitions. In *Proceedings of the 14th International Workshop and 15th International Conference on Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pp. 315–326. Springer-Verlag. ISBN 978-3-642-22934-3.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, {NIPS} 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pp. 849–856.

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, pp. 701–710, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623732. URL http://doi.acm.org/10.1145/2623330.2623732.

Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 459–467, Marina Del Rey CA USA, February 2018a. ACM. ISBN 978-1-4503-5581-0. doi: 10.1145/3159652.3159706. URL https://dl.acm.org/doi/10.1145/3159652.3159706.

Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings of the Eleventh {ACM} International Conference on Web Search and Data Mining, {WSDM} 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pp. 459–467, 2018b. doi: 10.1145/3159652.3159706. URL https://doi.org/10.1145/3159652.3159706.

Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000. ISSN 0162-8828. doi: 10.1109/34.868688. URL https://doi.org/10.1109/34.868688. Place: Washington, DC, USA Publisher: IEEE Computer Society.

G. W. Stewart and Ji-Guang Sun. *Matrix Perturbation Theory*. Academic Press, Boston, 1990.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077, Florence Italy, May 2015. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-3469-3. doi: 10.1145/2736277.2741093. URL https://dl.acm.org/doi/10.1145/2736277.2741093.

Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, November 2011. ISSN 1384-5810, 1573-756X. doi: 10.1007/s10618-010-0210-x. URL http://link.springer.com/10.1007/s10618-010-0210-x.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. VERSE: Versatile Graph Embeddings from Similarity Measures. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, pp. 539–548, Lyon, France, 2018. ACM Press. ISBN 978-1-4503-5639-8. doi: 10.1145/3178876.3186120. URL http://dl.acm.org/citation.cfm?doid=3178876.3186120.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Ulrike von Luxburg. A Tutorial on Spectral Clustering. *CoRR*, abs/0711.0, 2007. URL http://arxiv.org/abs/0711.0189.

A DATASETS AND HYPERPARAMETERS.

Datasets. We use 6 labeled datasets that have been used widely as benchmarks. These datasets have been processed to remove small connected components because spectral clustering naturally works on a connected graph[2]; self-loops are also removed. For all datasests, we treat the graphs as undirected and only consider the largest connected component. Table 2 summarizes the number of vertices, edges and classes of these largest connected components.

Dataset	CITESEER	CORA	WIKICS	PUBMED	ARXIV	PRODUCTS
$\lvert V\rvert$	3,327	2,708	11,701	19,717	169,343	2,449,029
$\lvert \tilde{V}\rvert$	2,120	2,485	11,311	19,717	169,343	2,385,902
$\lvert E\rvert$	4,614	6,632	215,863	44,326	2,315,598	123,718,152
$\lvert \tilde{E}\rvert$	3,679	5,069	215,554	44,324	2,315,598	123,612,606
#classes	6	7	10	3	40	47

Table 2: Number of vertices and edges in the original and processed graphs

Code and Experiments. All experiments were performed on Google Colab. The final paper will disseminate all code, including the models and the experimental setting.

Hyperparameter Settings. In RCS-CLASSIFY (algorithm 1), we take $k = 2 \cdot nc$, where nc is the number of classes. In non-reported experiments we have found that further increasing the number of dimensions can slightly increase accuracy, but we take a theory-consistent decision to make the dimension proportional to number of clusters. The Random Walker algorithm does not have any hyperparameters or learnable parameters. For the GCN model we used the recommended settings, using the identity matrix in the input into the GCN. For NetMF we followed the recommended default settings, using an embedding of 128 dimensions. The dimension for DeepWalk is set to be $2 \cdot nc$; this deviates from the recommendation but it is added here for direct comparison with the setting for RCS-CLASSIFY.

B FURTHER DISCUSSION

• We have also considered the applications of Support Vector Machines on the radially projected spectral embedding. The results were not competitive, and thus they are not included here.

• The RANDOMWALKER method has rarely -if ever- been used as benchmark for supervised node classification on social graphs. However the success of C&S, a closely related algorithm, motivates the experimental study of the RANDOMWALKER algorithm, the strengths of which include the fact that it does not have any hyperparameters or trainable parameters. We see that in multiple cases RANDOMWALKER is a viable alternative to other models, especially considering that with a proper implementation using Laplacian solvers, RANDOMWALKER is **by far faster** than all other methods in Table 1. The C&S process is not applicable to the output of RANDOMWALKER for certain concrete technical reasons, but a worthy research topic it to modify RANDOMWALKER in a way that would enable the application of C&S, in order to enhance its performance.

[2]One can use the node features to connect these smaller connected components. We avoid this practice in order to stick with the decision to only use graphic data.