Interpretable Clustering on Dynamic Graphs with Recurrent Graph Neural Networks

Yuhang Yao, Carlee Joe-Wong
Carnegie Mellon University
{yuhangya,cjoewong}@andrew.cmu.edu

Abstract

We study the problem of clustering nodes in a dynamic graph, where the connections between nodes and nodes' cluster memberships may change over time, e.g., due to community migration. We first propose a dynamic stochastic block model that captures these changes, and a simple decay-based clustering algorithm that clusters nodes based on weighted connections between them, where the weight decreases at a fixed rate over time. This decay rate can then be interpreted as signifying the importance of including historical connection information in the clustering. However, the optimal decay rate may differ for clusters with different rates of turnover. We characterize the optimal decay rate for each cluster and propose a clustering method that achieves almost exact recovery of the true clusters. We then demonstrate the efficacy of our clustering algorithm with optimized decay rates on simulated graph data. Recurrent neural networks (RNNs), a popular algorithm for sequence learning, use a similar decay-based method, and we use this insight to propose two new RNN-GCN (graph convolutional network) architectures for semi-supervised graph clustering. We finally demonstrate that the proposed architectures perform well on real data compared to state-of-the-art graph clustering algorithms.

Introduction

Clustering nodes based on their connections to each other is a common goal of analyzing graphs, with applications ranging from social to biological to logistics networks. Most such clustering approaches assume that the connections (i.e., edges) between nodes, and thus the optimal clusters, do not change over time (Lei, Rinaldo et al. 2015; Qin and Rohe 2013). In practice, however, many graph structures will evolve over time. Users in social networks, for example, may migrate from one community to another as their interests or employment status changes, forming new connections with other users (i.e., new edges in the graph) and changing the cluster or community to which they belong. Thus, clustering algorithms on such evolving graphs should be able to track changes in cluster membership over time.

A major challenge in tracking cluster membership changes is to carefully handle historical information and assess its value in predicting the current cluster membership. Since clusters will often evolve relatively slowly, an extreme approach that does not consider edges formed in the past risks ignoring useful information about the majority of nodes whose memberships have not changed. On the other hand, making no distinction between historical and more recently formed edges may lead to slow detection of nodes' membership changes, as the historically formed edges would dominate until the nodes have enough time to make connections within their new clusters. Prior works have balanced these effects by introducing a *decay rate*: the weight of each edge is reduced by a constant decay factor in each time step between the connection formation and the time at which cluster membership is estimated. The cluster membership can then be estimated at any given time, by taking the weighted connections as input to a static algorithm like the well-studied spectral clustering (Abbe 2017).

Accounting for historical node connections with a single decay rate parameter offers the advantage of interpretability: the decay rate quantifies the emphasis put on historically formed edges, which can be tuned for specific datasets. Yet while prior works have examined the optimal decay rate for stylized network models, they use a single decay rate for all edges (Keriven and Vaiter 2020). In practice, the optimal rate will likely vary, e.g., with higher decay rates for clusters with higher membership turnover where historical information might reflect outdated cluster memberships, making it less useful. Introducing different decay rates for each cluster, on the other hand, raises a new challenge: since we do not know the true cluster memberships for each node, we may use the wrong decay rate if a node is erroneously labeled. Moreover, the optimal decay rates for different clusters will be correlated due to connections between nodes in different clusters that themselves must be optimally weighted, potentially making the decay rates difficult to optimize.

More sophisticated semi-supervised clustering methods combine LSTM (long-short-term memory) or RNN (recurrent neural network) structures with graph convolutional networks (GCNs), producing a neural network that classifies nodes based on their cluster membership labels. This network can be carefully trained to optimize the use of historical edge information, without explicitly specifying different node decay rates (Pareja et al. 2020). However, while such algorithms show impressive empirical performance on large graph datasets, they are generally not easily interpretable.

Our work seeks to connect the theoretical analysis of graph clustering algorithms with the graph neural networks commonly used in practice. Our key insight in doing so is that *prefacing a GCN with a RNN layer* can be interpreted as imposing a decay rate on node connections that depends on each node's current cluster membership, and then approximating spectral clustering on the resulting weighted graph via the GCN. Following this insight, we propose two new *transitional RNN-GCN neural network architectures* (RNNGCN and TRNNGCN) for semi-supervised clustering. We derive the theoretically optimal decay rates for nodes in each cluster under stylized graph models, and show that the weights learned for the RNN layer in TRNNGCN qualitatively match the theoretically optimal ones. After reviewing related work on theoretical and empirical graph clustering, we make the following specific contributions:

- A **theoretical analysis of the optimal decay rates** for spectral clustering algorithms applied to the dynamic stochastic block model, a common model of graph clustering dynamics (Keriven and Vaiter 2020).
- Two **new RNN-GCN neural network architectures** that use an interpretable RNN layer to capture the dynamics of evolving graphs and GCN layers to cluster the nodes.
- Our algorithm can achieve **almost exact recovery** by including a RNN layer that decays historical edge information. Static methods can only partially recover the true clusters when nodes change their cluster memberships with probability $\mathcal{O}\left(\frac{\log n}{n}\right)$, n being the number of nodes.
- **Experimental results** on real and simulated datasets that show our proposed RNN-GCN architectures outperform state-of-the-art graph clustering algorithms.

Related Work

Over the past few years, there has been significant research dedicated to graph clustering algorithms, motivated by applications such as community detection in social networks. While some works consider theoretical analysis of such clustering algorithms, more recently representation learning algorithms have been proposed that perform well in practice with few theoretical guarantees. We aim to *connect* these approaches by using a theoretical analysis of decay-based dynamic clustering algorithms to design a new neural network-based approach that is easily interpretable.

Theoretical analyses. Traditional spectral clustering algorithms use the spectrum of the graph adjacency matrix to generate a compact representation of the graph connectivity (Lei, Rinaldo et al. 2015; Qin and Rohe 2013). A line of work on static clustering algorithms uses the stochastic block model for graph connectivity (Abbe 2017), which more recent works have extended to a dynamic stochastic block model (Keriven and Vaiter 2020; Pensky, Zhang et al. 2019). While these works do not distinguish between clusters with different transition probabilities, earlier models incorporate such heterogeneity (Xu 2015). Other works use a Bayesian approach (Yang et al. 2011), scoring metrics (Agarwal et al. 2018; Zhuang, Chang, and Li 2019), or multi-armed bandits (Mandaglio and Tagarelli 2019) to detect communities and their evolution, while Xu, Kliger, and Hero (2010) use a decay rate similar to the one we propose.

As **representation learning** becomes popular, graph neural networks (Zhang and Chen 2018; Wu et al. 2020; Kipf and Welling 2016) such as GraphSage (Hamilton, Ying, and Leskovec 2017) have been used to cluster nodes in graphs based on (static) connections between nodes and node features. Graph Attention Networks (GAT) (Veličković et al. 2017; Xu et al. 2019b) use attention-based methods to construct a neural network that highlights the relative importance of each feature, while dynamic supervised (Kumar, Zhang, and Leskovec 2019) and unsupervised (Goyal, Chhetri, and Canedo 2020) methods can track general network dynamics, or may be designed for clustering on graphs with dynamic edges and dynamic node features (Chen et al. 2018; Xu et al. 2019a, 2020). EvolveGCN (Pareja et al. 2020) usesa GCN to evolve the RNN weights, which is similar to our approach; however, we ensure interpretability of the RNN weights by placing the RNN before the GCN layers, which we show improves the clustering performance.

Finally, several works have considered the **interpretability of general GCN and RNN structures** (Dehmamy, Barabási, and Yu 2019; Liang et al. 2017; Guo, Lin, and Antulov-Fantulin 2019), such as GNNExplainer (Ying et al. 2019). In the context of graph clustering, some works have used attention mechanisms to provide interpretable weights on node features (Xu et al. 2019a), but attention may not capture true feature importance (Serrano and Smith 2019). Moreover, these works do not consider the importance of *historical* information, as we consider in this work.

Model

We first introduce a dynamic version of the Stochastic Block Model (SBM) often used to study graph clustering (Holland, Laskey, and Leinhardt 1983; Abbe 2017), which we will use for our theoretical analysis in the rest of the paper.

Stochastic Block Model

For positive integers K and n, a probability vector $p \in [0,1]^K$, and a symmetric connectivity matrix $B \in [0,1]^{K\times K}$, the SBM defines a random graph with n nodes split into K clusters. The goal of a prediction method for the SBM is to correctly divide nodes into their corresponding clusters, based on the graph structure. Each node is independently and randomly assigned a cluster in $\{1,...,K\}$ according to the distribution p; we can then say that a node is a "member" of this cluster. Undirected edges are independently created between any pair of nodes in clusters i and j with probability B_{ij}, where the (i,j) entry of B is

$$B_{ij} = \begin{cases} \alpha, & i = j \\ \tau\alpha, & i \neq j, \end{cases} \tag{1}$$

for $\alpha \in (0,1)$ and $\tau \in (0,1)$, implying that the probability of an edge forming between nodes in the same cluster is α (which is the same for each cluster) and the edge formation probability between nodes in different clusters is $\tau\alpha$.

Let $\Theta \in \{0,1\}^{n\times K}$ denotes the matrix representing the nodes' cluster memberships, where $\Theta_{ik} = 1$ indicates that node i belongs to the k-th cluster, and is 0 otherwise. We use $A \in \{0,1\}^{n\times n}$ to denote the (symmetric) adjacency matrix of the graph, where A_{ij} indicates whether there is a connection (edge) between node i and node j. From our node connectivity model, we find that given Θ, for $i < j$, we have

$$A_{ij}|\{\Theta_{ik} = 1, \Theta_{jl} = 1\} \backsim \text{Ber}(B_{kl}), \tag{2}$$

where Ber(p) indicates a Bernoulli random variable with parameter p. We define $A_{ii} = 0$ (nodes are not connected directly to themselves) and since all edges are undirected, $A_{ij} = A_{ji}$. We further define the connection probability matrix $P = \Theta B \Theta^T \in [0,1]^{n\times n}$, where P_{ij} is the connection probability of node i and node j and $\mathbb{E}[A] = P - \text{diag}(P)$.

Dynamic Stochastic Block Model

We now extend the SBM model to include how the graph evolves over time. We consider a set of discrete time steps $t = 1, 2, \ldots, T$. At each time step t, the Dynamic SBM generates new intra- and inter-cluster edges according to the probabilities α and $\tau\alpha$ as defined for the SBM above. All edges persist over time. We assume a constant number of nodes n, number of clusters K, and connectivity matrix B, but the node membership matrix Θ_t depends on time t, i.e., nodes' cluster memberships may change over time. We similarly define the connectivity matrix $P_t = \Theta_t B (\Theta_t)^T$.

We model changes in nodes' cluster memberships as a Markov process with a constant transition probability matrix $H \in [0,1]^{K\times K}$. Let $\varepsilon_j \in (0,1)$ denotes the change probability of nodes in cluster j, i.e., the probability a node in cluster j changes its membership. At each time step, node v_i in cluster j changes its membership to cluster k with the following probability (independently from other nodes):

$$H_{j,k} = \mathbb{P}\left[\Theta_{ik}^t = 1 | \Theta_{ij}^{t-1} = 1\right] = \begin{cases} 1 - \varepsilon_j, & j = k \\ \dfrac{\varepsilon_j}{K-1}, & j \neq k, \end{cases}$$

Note that ε_j may be specific to cluster j, e.g., if some clusters experience less membership turnover. We give an example of such a graph in our experimental evaluation. The goal of a clustering algorithm on a graph is to recover the membership matrix Θ up to column permutation. Static clustering algorithms give an estimate $\hat{\Theta}$ of the node membership; a dynamic clustering algorithm should produce such an estimate for each time t. We define two performance metrics for these estimates (in dynamic graphs, they may be evaluated for an estimate $\hat{\Theta} = \hat{\Theta}_t$ relative to $\Theta = \Theta_t$ at any time t):

Definition 1 (Relative error of $\hat{\Theta}$). The relative error of a clustering estimate $\hat{\Theta}$ is

$$E(\hat{\Theta}, \Theta) = \frac{1}{n} \min_{\pi \in \mathcal{P}} \|\hat{\Theta}\pi - \Theta\|_0, \tag{3}$$

where $\mathcal{P}$ is the set of all $K \times K$ permutation matrices and $\|.\|_0$ counts the number of non-zero elements of a matrix.

Definition 2 (Almost Exact Recovery). A clustering estimate $\hat{\Theta}$ achieves almost exact recovery when

$$\mathbb{P}\left[1 - \frac{1}{n}\min_{\pi\in\mathcal{P}} \|\hat{\Theta}\pi - \Theta\|_0 = 1 - o(1)\right] = 1 - o(1). \tag{4}$$

which also implies that the expectation of $E(\hat{\Theta}, \Theta)$ is $o(1)$.

Our goal is then to find an algorithm that produces an estimate $\hat{\Theta}$ minimizing $E(\hat{\Theta}, \Theta)$. In the next section, we discuss the well-known (static) spectral clustering algorithm and analyze a simple decay-based method that allows a static algorithm to make dynamic membership estimates.

Spectral Clustering with Decay Rates

We now introduce the Spectral Clustering algorithm and optimize the decay rates to minimize its relative error.

Spectral Clustering Algorithm

Spectral Clustering is a commonly used unsupervised method for graph clustering. The key idea is to apply K-means clustering to the K-leading left singular vectors of the adjacency matrix A (Stella and Shi 2003); we denote the corresponding matrix of singular vectors as E_K. We then estimate the membership matrix $\bar{\Theta}$ by solving

$$(\bar{\Theta}, \bar{C}) \in \underset{\Theta\in\{0,1\}^{n\times K}, C\in\mathbb{R}^{K\times K}}{\arg\min} \|\Theta C - E_K\|_F^2, \tag{5}$$

where $\|.\|_F$ denotes the Frobenius norm. It is well known that finding a global minimizer of Eq. (5) is NP-hard. However, efficient algorithms (Kumar, Sabharwal, and Sen 2004) can find a $(1+\delta)$-approximate solution $(\hat{\Theta}, \hat{C})$, i.e., with $\|\hat{\Theta}\hat{C} - E_K\|_F^2 \leq (1+\delta)\|\bar{\Theta}\bar{C} - E_K\|_F^2$.

Introducing Decay Rates

Spectral clustering performs poorly on the dynamic SBM:

Proposition 1 (Partial Recovery of Spectral Clustering). When nodes change their cluster membership over time with probabilities $\varepsilon_j = \mathcal{O}(\frac{\log n}{n})$, Spectral Clustering recovers the true clusters at time T with relative error $E(\hat{\Theta}_T, \Theta_T) = \mathcal{O}(\frac{\log n}{n}T)$.

To improve this performance, one can replace the adjacency matrix A_t with an exponentially smoothed version $\hat{A}_t$:

$$\hat{A}_t = (1-\lambda)\hat{A}_{t-1} + \lambda A_t, \tag{6}$$

where we call $\lambda \in [0,1]$ the *decay rate* (Chi et al. 2009). Intuitively, a larger value of λ puts less weight on the past information, "forgetting" it faster. However, in the dynamic SBM, each cluster j may have a different change probability ε_j, implying that they may benefit from using different decay rates λ. We thus introduce a decay matrix $\Lambda \in [0,1]^{K\times K}$ that gives a different decay rate to connections between each pair of clusters:

$$\hat{A}_t = (1 - \Theta_t\Lambda(\Theta_t)^T) \odot \hat{A}_{t-1} + \Theta_t\Lambda(\Theta_t)^T \odot A_t. \tag{7}$$

Bounding the Relative Error

Our analysis uses Lei, Rinaldo et al. (2015)'s result that the relative error rate of the Spectral Clustering on the dynamic SBM at each time t is bounded by the concentration of the adjacency matrix around its expectation:

$$E(\hat{\Theta}, \Theta) \lesssim (1+\delta)\frac{n'_{\max}K}{n\alpha^2 n_{\min}^2 \tau^2}\|\hat{A} - P\|^2, \tag{8}$$

where $n'_{\max}$ and $n_{\min}$ are respectively the second largest and smallest cluster sizes, and $\|.\|$ denotes the spectral norm.

Thus, $E(\hat{\Theta}, \Theta)$ is determined by the concentration $\|\hat{A} - P\|$, where $P = P_t = \Theta_t B(\Theta_t)^T$ as defined in the SBM model. To bound this concentration, we consider K diagonal blocks of the adjacency matrix $\hat{A}_t$, with each block corresponding to edges between nodes in a single cluster, after re-indexing the nodes as necessary. Let $\hat{A}_t^k$ denote the block matrix corresponding to cluster k, and similarly consider K blocks P_k^t of the connection probability matrix P_t. We can then upper-bound $\left\|\hat{A}_t^k - P_t^k\right\|$ in terms of the decay rate:

Proposition 2 (Optimal Decay Rate)**.** The concentration of each block k is upper-bounded by

$$\left\|\hat{A}_t^k - P_t^k\right\| \lesssim E_1(\beta_k) + E_2(\beta_k), \tag{9}$$

where β_k denotes the maximum decay rate of class k and

$$E_1(\beta_k) = \sqrt{n\alpha\beta_k},\ E_2(\beta_k) = \alpha\sqrt{\frac{n^2\varepsilon_k}{\beta_k}}, \tag{10}$$

which is minimized when $\beta_k = \sqrt{n\alpha\varepsilon_k}$.

We formally prove this result in our supplementary material. The intuition is that if the change probability ε_k is larger, we need a higher decay rate to remember less past information. We thus define the decay rates as

$$\Lambda_{jk} = \begin{cases} \min(1, \sqrt{n\alpha\varepsilon_k}), & j = k \\ 1, & j \neq k. \end{cases} \tag{11}$$

This decay rate yields almost exact recovery:

Proposition 3 (Almost Exact Recovery)**.** Let $\lambda_{\max}$ denote the maximum element on the diagonal of Λ. With probability at least $1 - n^{-\nu}$ for any $\nu > 0$, at any time t we have

$$\left\|\hat{A}_t - P_t\right\| \lesssim \sqrt{n\alpha\lambda_{\max}} \tag{12}$$

When K is constant, $\varepsilon_k = \mathcal{O}\left(\frac{\log n}{n}\right)$ and $\alpha = \mathcal{O}\left(\frac{\log n}{n}\right)$, the relative error is $\mathcal{O}\left(\frac{1}{n^{\frac{1}{4}}\log n}\right)$, which implies almost exact recovery at time T.

Connection between GCN and Spectral Clustering

We empirically demonstrate that Proposition 2's decay rate is optimal by varying the decay rates used in both spectral clustering and the commonly used Graph Convolutional Network (GCN), which is a first-order approximation of spectral convolutions on graphs (Kipf and Welling 2016). A multi-layer GCN has the layer-wise propagation rule:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}), \tag{13}$$

where $\widetilde{A} = A + I_N$, I_N is the identity matrix, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$ and $W^{(l)}$ is a layer-specific trainable weight matrix. The activation function is σ, typically ReLU (rectified linear units), with a softmax in the last layer for graph clustering. The node embedding matrix in the l-th layer is $H^{(l)} \in \mathbb{R}^{N\times D}$,

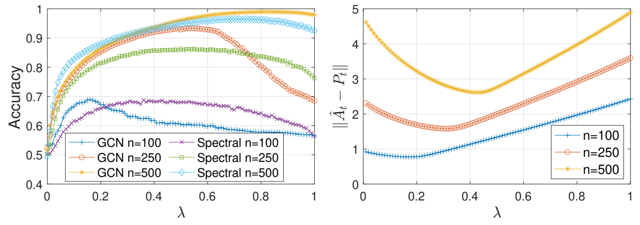

Figure 1: Accuracy and Spectral Norm as we vary n. The optimal decay rate λ increases with n, as in Proposition 2.

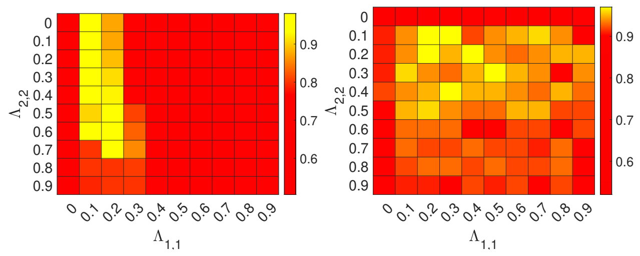

Figure 2: Accuracy as we vary $\Lambda_{1,1}$ and $\Lambda_{2,2}$. Spectral Clustering(left) and GCN(right) have similar optimal decay matrix. The change probabilities are $\varepsilon_1 = 0.05, \varepsilon_2 = 0.1$.

which contains high-level representations of the graph nodes transformed from the initial features; $H^{(0)} = I_N$. Figure 1 shows that Spectral Clustering and GCN have qualitatively similar accuracy on simulated data as we vary the decay rate λ (the same λ is used for all nodes). As expected from Eq. (8) and Proposition 2, the optimal decay rate λ increases as we increase the number of nodes n, as does the value of λ that minimizes the spectral norm $\|\hat{A} - P\|$. Although the optimal decay rate is consistently larger than the one minimizing the spectral norm (which upper-bounds the relative error as in Eq. (8)), GCN's accuracy is more correlated with the spectral norm, which is the first singular value of the smoothed adjacency matrix.

We then perform a grid search for the optimal decay matrix Λ on simulated data with $n = 200$ nodes and change probabilities $\varepsilon_1 = 0.05$ and $\varepsilon_2 = 0.1$. As shown in Figure 2, GCN and Spectral Clustering achieve high accuracy. Cluster 2, which has a higher ε_2, has larger decay rate $\Lambda_{2,2}$, as expected from Proposition 2, for both GCN and Spectral Clustering.

Decay Rates as RNNs

Although searching for the optimal decay matrix in Spectral Clustering can result in good performance, this method is expensive: the grid search for the optimal decay matrix can be time-consuming, and the time complexity of calculating the spectral norm is $\mathcal{O}(n^3)$. In this section, we propose two neural network architectures, RNNGCN and TRNNGCN, that use a single decay rate λ and decay matrix Λ, respectively, and then show they perform well on simulated data.

RNNGCN The RNNGCN model uses a single decay rate $\lambda \in [0, 1]$ as the RNN parameter. RNNGCN first uses a Recurrent Neural Network to learn the decay rate, then uses a

Algorithm 1: RNNGCN

```
Input  : Temporal graph $(A_1, A_2, ..., A_T)$,
         Membership matrix of training data $\Theta_T^{train}$
Output: Membership matrix estimate $\hat{\Theta}_T$
1  $\hat{A} = A_0, H_0 = I_N$;
2  for iteration $i = 1, ..., I$ do
3      for $t = 2, ..., T$ do
4          $\hat{A}_t = (1 - \lambda)\hat{A}_{t-1} + \lambda A_t$
5      end
6      $H^{(1)} = \sigma_1(\hat{A}_T H^{(0)} W^{(1)})$
7      $H^{(2)} = \sigma_2(\hat{A}_T H^{(1)} W^{(2)})$
8      CrossEntropyLoss($H_i^{train}$, $\Theta_T^{train}$)
9      Backward()
10 end
11 $\hat{\Theta}_T = \text{Onehot}(\arg\max_{1 \le j \le n} H_{jk}^{(2)})$
```

two-layer GCN to cluster the weighted graphs. The formal model is shown in Algorithm 1, where σ_1 denotes a ReLU layer and σ_2 is a Softmax layer.

Transitional RNNGCN (TRNNGCN) The TRNNGCN network is similar to the RNNGCN, but uses a matrix $\Lambda \in [0, 1]^{K \times K}$ to learn the decay rates for different pairs of classes. During the training process, the labels (cluster memberships) of the training nodes are known while the labels of other nodes remain unknown, so we use the cluster prediction $\hat{\Theta}_{i-1}$ from each iteration $i - 1$ to determine the decay rates for each node in iteration i. The TRNNGCN model replaces the decay method (line 4 of Algorithm 1) with

$$\hat{A}_t = (1 - \hat{\Theta}_{i-1}\Lambda(\hat{\Theta}_{i-1})^T) \odot \hat{A}_{t-1} + \hat{\Theta}_{i-1}\Lambda(\hat{\Theta}_{i-1})^T \odot \hat{A}_t,$$

where $\odot$ denotes element-wise multiplication. After each iteration, it calculates $\hat{\Theta}_i$ as the input of the next iteration.

Empirical Validation We validate the performance of RNNGCN and TRNNGCN on data generated by the dynamic stochastic block model. Our graph has 200 nodes, 23190 edges, 50 time steps and 2 clusters. The probabilities of forming an edge between two nodes of the same or different clusters are $\alpha = 0.02$ and $\tau\alpha = 0.001$, respectively, and a node changes its cluster membership with probability $\varepsilon_1 = 0.05$ and $\varepsilon_2 = 0.1$ for clusters 1 and 2 respectively.

Figure 3 compares the RNNGCN and TRNNGCN performance with the static Spectral Clustering and GCN methods. For better visualization, the value at each time step is averaged with the 2 timesteps immediately before and after. The performance of GCN and Spectral Clustering decreases over time, as in later timesteps they use accumulated historical information that may no longer be relevant. RNNGCN and TRNNGCN show consistently high performance over time, indicating that they optimally utilize historical information. On average over time, TRNNGCN leads to 5% accuracy and AUC (area under the ROC curve) improvement, and a 10% higher F1-score, than RNNGCN, due to using a lower decay rate for the class with smaller change probability.

Experiments

In this section, we validate the performance of RNNGCN and TRNNGCN on real datasets, compared to state-of-the-art baselines. We first describe the datasets used and the baselines considered, and then present our results.

Datasets

We conducted experiments on five real datasets, as shown in Table 1, which have the properties shown in Table 2. All datasets have edges that form at different times, although only nodes in DBLP-E change their class (cluster membership) over time. We include four datasets with separate, time-varying features associated with each node (DBLP-3, DBLP-5, Brain and Reddit) to test RNNGCN's and TRNNGCN's ability to generalize to datasets with node features.

Dataset	Nodes	Edges	Time Steps	Classes
DBLP-E	6942	327392	14	2
DBLP-3	4257	23540	10	3
DBLP-5	6606	42815	10	5
Brain	5000	1955488	12	10
Reddit	8291	264050	10	4

Table 1: Real datasets used to evaluate our methods.

Dataset	Dynamic Edge	Dynamic Class	Features
DBLP-E	$\surd$	$\surd$	$\times$
DBLP-3	$\surd$	$\times$	100
DBLP-5	$\surd$	$\times$	100
Brain	$\surd$	$\times$	20
Reddit	$\surd$	$\times$	20

Table 2: Properties of datasets in Table 1.

DBLP-E dataset is extracted from the computer science bibliography website DBLP[1], which provides open bibliographic information on major computer science journals and conferences. Nodes represent authors, and edges represent co-authorship from 2004 to 2018. Each year is equivalent to one timestep, and co-author edges are added in the year a coauthored paper is published. Labels represent the author research area ("computer networks" or "machine learning") and may change as authors switch their research focus.

DBLP-3 & DBLP-5 use the same node and edge definitions as DBLP-E, but also include node features extracted by `word2vec` (Mikolov et al. 2013) from the authors' paper titles and abstracts. The authors in DBLP-3 and DBLP-5 are clustered into three and five classes (research areas) respectively that do not change over time.

Reddit dataset is generated from Reddit[2], a social news aggregation and discussion website. The nodes represent posts and two posts are connected if they share keywords. Node features are generated by `word2vec` on the post comments (Hamilton, Ying, and Leskovec 2017).

[1]https://dblp.org

[2]https://www.reddit.com/

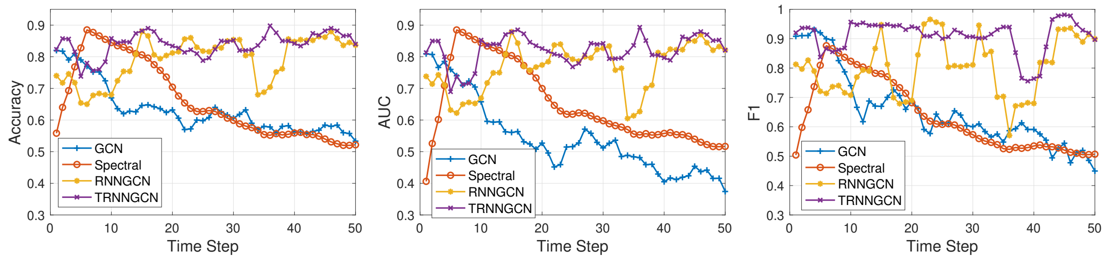

Figure 3: On simulated data with heterogeneous cluster transition probabilities, TRNNGCN and RNNGCN, which use optimized decay rates to account for historical information, outperform the static GCN and Spectral Clustering methods. TRNNGCN slightly outperforms RNNGCN.

Brain dataset is generated from functional magnetic resonance imaging (fMRI) data[3]. Nodes represent cubes of brain tissue, and two nodes are connected if they show similar degrees of activation during the time period. Node features are generated by principal component analysis on the fMRI.

Baselines and Metrics

We compare our RNNGCN and TRNNGCN with multiple baselines. GCN, GAT (Veličković et al. 2017) and GraphSage (Hamilton, Ying, and Leskovec 2017) are supervised methods that include node features, while Spectral Clustering is unsupervised without features; all of these methods ignore temporal information. DynAERNN (Goyal, Chhetri, and Canedo 2020) is an unsupervised method, and GCNLSTM (Chen et al. 2018) and EGCN (Pareja et al. 2020) are supervised methods, which all utilize temporal information of both graphs and features. We evaluate the performance of methods with the standard accuracy (ACC), area under the ROC curve (AUC) and F1-score classification metrics.

Experiment Settings

We divide each dataset into 70% training/ 20% validation/ 10% test points. Each method uses two hidden Graph Neural Network layers (GCN, GAT, GraphSage, etc.) with the layer size equal to the number of classes in the dataset. We add a dropout layer between the two layers with dropout rate 0.5. We use the Adam optimizer with learning rate 0.0025. Each method is trained with 500 iterations.

For static methods (GCN, GAT, GraphSage and Spectral Clustering) we first accumulate the adjacency matrices of graphs at each time step, then cluster on the normalized accumulated matrix. DynAERNN, GCNLSTM, and EGCN use the temporal graphs and temporal node features as input. For our RNNGCN and TRNNGCN, we use the temporal graphs and the node features at the last time step as input. The code of all methods and datasets are publicly available[4].

Experimental Results

Node Classification with Temporal Labels We first compare the predictions of temporally changing labels in DBLP-E. Figure 4 shows the number of authors in the Machine

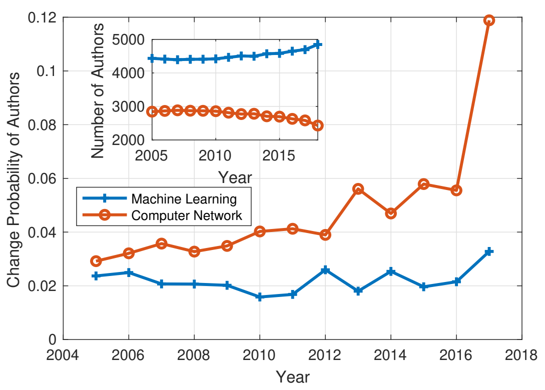

Figure 4: The two classes in DBLP-E exhibit different change probabilities, with computer network authors more likely to change their labels to machine learning. This trend accelerates after 2014.

Learning and Computer Network fields in the years 2004-2018, as well as the probabilities that authors in each class change their labels. We observe that (i) the classes have different change probabilities (with users more likely to move from computer networks to machine learning) and (ii) the change probabilities evolve over time, with more users migrating to machine learning since 2013. This dataset thus allows us to test RNNGCN's and TRNNGCN's abilities to adapt the optimal decay rate for each class.

Figure 5 shows the performance of RNNGCN and TRNNGCN in DBLP-E. Similar to Figure 3's result on the simulated data, GCN's performance decreases over time as the accumulated effect of class change increases. Spectral Clustering consistently performs poorly since it cannot learn the high-dimensional patterns of the DBLP-E graph. RNNGCN and TRNNGCN maintain good performance and fully utilize the temporal information. As the two classes have different change probabilities, TRNNGCN learns a better decay rate and performs better. We further show the average accuracy, AUC, and F1-score for each baseline method at each timestep; RNNGCN and TRNNGCN consistently outperform the other baselines. GCNLSTM comes the closest to matching their performance. GCNLSTM uses a LSTM layer to account for historical information, which is similar

[3]https://tinyurl.com/y4hhw8ro

[4]https://github.com/InterpretableClustering/InterpretableClustering

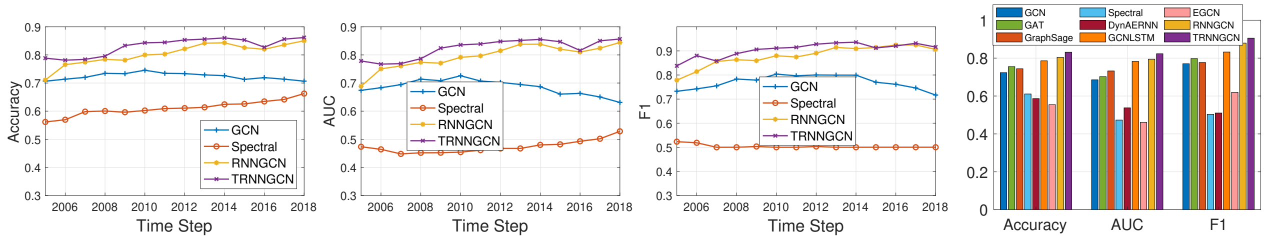

Figure 5: On DBLP-E data, TRNNGCN and RNNGCN outperform the static GCN and spectral clustering methods and show better performance over time, indicating that they can optimize their decay rates to account for historical information. TRNNGCN slightly outperforms RNNGCN.

	DBLP-3			DBLP-5			Reddit			Brain		
	ACC	AUC	F1	ACC	AUC	F1	ACC	AUC	F1	ACC	AUC	F1
GCN	71.6	62.2	35.8	64.9	51.0	***58.7***	31.0	24.5	47.4	35.2	25.0	80.3
GAT	70.9	59.4	57.8	62.3	48.2	51.4	16.8	4.8	50.0	34.6	26.4	81.6
GraphSage	74.5	63.6	55.0	***66.5***	53.9	55.1	29.2	20.7	42.5	**44.2**	***41.9***	**86.7**
Spectral	45.7	51.6	51.2	43.8	45.6	51.3	30.1	24.1	51.7	42.7	41.7	68.1
DynAERNN	48.1	54.2	50.8	33.1	39.1	51.2	31.1	**31.7**	**54.1**	20.5	20.3	55.6
GCNLSTM	74.5	63.6	48.4	***66.5***	53.2	54.6	31.9	25.5	46.1	38.8	32.9	***85.9***
EGCN	72.3	60.7	48.1	63.2	50.6	53.2	28.3	12.5	50.0	28.6	26.1	73.7
RNNGCN	***75.9***	***68.0***	***66.7***	65.7	***55.4***	58.6	**33.6**	20.5	49.7	41.0	38.6	84.7
TRNNGCN	**78.0**	**72.1**	**73.8**	**67.4**	**57.9**	**63.5**	**33.6**	***25.6***	***53.2***	***43.8***	**42.4**	85.7

Table 3: TRNNGCN consistently achieves the best (bold) or second-best (bold italics) accuracy (ACC), area under the ROC curve (AUC), and F1 score compared to baseline algorithms on four temporal datasets.

to our methods but lacks interpretability as the LSTM operates on the output of the GCN layer (which is not readily interpretable) instead of the original graph adjacency information. We use a RNN instead of LSTM layer in our algorithms for computational efficiency.

Node Classification with Temporal Features Although our analysis is based on dynamic networks without features, Table 3's performance results demonstrate the applicability of our RNNGCN and TRNNGCN algorithms to the DBLP-3, DBLP-5, Brain, and Reddit datasets with node features. TRNNGCN achieves the best or second-best performance consistently on all datasets, even outperforming EGCN and GCNLSTM, which unlike TRNNGCN fully utilize the historical information of node features. While GraphSAGE shows good accuracy, AUC, and F1-score on the Brain dataset, no other baseline method does well across all three metrics for any other dataset. RNNGCN performs second-best on DBLP-3 but worse on the other datasets, likely because those datasets have more than three classes, which would likely have different optimal decay rates. TRNNGCN can account for these differences, but RNNGCN cannot.

We further highlight the importance of taking into account historical information by noting that the static baselines (GCN, GAT, GraphSAGE, and spectral clustering) generally perform poorly compared to the dynamic baselines (DynAERNN, GCNLSTM, EGCN). DynAERNN can perform significantly worse than GCNLSTM and EGCN, likely because it is an unsupervised method that cannot take advantage of labeled training data. Thus, RNNGCN and TRNNGCN's good performance is likely due to their ability to optimally take advantage of historical graph information, even if they cannot use historical node feature information.

Conclusion

This work proposes RNNGCN and TRNNGCN, two new neural network architectures for clustering on dynamic graphs. These methods are inspired by the insight that RNNs progressively decrease the weight placed on their inputs over time according to a learned decay rate parameter. This decay rate can in turn be interpreted as the importance of historical connection information associated with each community or cluster in the graph. We show that decaying historical connection information can achieve almost exact recovery when used for spectral clustering on dynamic stochastic block models, and that the RNN decay rates on simulated data match the theoretically optimal decay rates for such stochastic block models. We finally validate the performance of RNNGCN and TRNNGCN on a range of real datasets, showing that TRNNGCN consistently outperforms static clustering methods as well as previously proposed dynamic clustering methods. This performance is particularly remarkable compared to dynamic clustering methods that account for historical information of both the connections between nodes and the node features; TRNNGCN ignores the historical feature information. We plan to investigate neural network architectures that reveal the importance of these dynamic node features in our future work. Much work also remains on better establishing the models' interpretability.

Acknowledgements

This research was partially supported by NSF grant CNS-1909306. The authors would like to thank Jinhang Zuo, Mengqiu Teng and Xiao Zeng for their inputs to the work.

References

Abbe, E. 2017. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research* 18(1): 6446–6531.

Agarwal, P.; Verma, R.; Agarwal, A.; and Chakraborty, T. 2018. DyPerm: Maximizing permanence for dynamic community detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 437–449. Springer.

Chen, J.; Xu, X.; Wu, Y.; and Zheng, H. 2018. Gc-lstm: Graph convolution embedded lstm for dynamic link prediction. *arXiv preprint arXiv:1812.04206* .

Chi, Y.; Song, X.; Zhou, D.; Hino, K.; and Tseng, B. L. 2009. On evolutionary spectral clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3(4): 1–30.

Dehmamy, N.; Barabási, A.-L.; and Yu, R. 2019. Understanding the representation power of graph neural networks in learning graph topology. In *Advances in Neural Information Processing Systems*, 15413–15423.

Goyal, P.; Chhetri, S. R.; and Canedo, A. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems* 187: 104816.

Guo, T.; Lin, T.; and Antulov-Fantulin, N. 2019. Exploring interpretable lstm neural networks over multi-variable data. *arXiv preprint arXiv:1905.12034* .

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 1024–1034.

Holland, P. W.; Laskey, K. B.; and Leinhardt, S. 1983. Stochastic blockmodels: First steps. *Social networks* 5(2): 109–137.

Keriven, N.; and Vaiter, S. 2020. Sparse and Smooth: improved guarantees for Spectral Clustering in the Dynamic Stochastic Block Model.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* .

Kumar, A.; Sabharwal, Y.; and Sen, S. 2004. A simple linear time (1+/spl epsiv/)-approximation algorithm for k-means clustering in any dimensions. In *45th Annual IEEE Symposium on Foundations of Computer Science*, 454–462. IEEE.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1269–1278.

Lei, J.; Rinaldo, A.; et al. 2015. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics* 43(1): 215–237.

Liang, X.; Lin, L.; Shen, X.; Feng, J.; Yan, S.; and Xing, E. P. 2017. Interpretable structure-evolving lstm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1010–1019.

Mandaglio, D.; and Tagarelli, A. 2019. Dynamic consensus community detection and combinatorial multi-armed bandit. In *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 184–187. IEEE.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; Schardl, T. B.; and Leiserson, C. E. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *AAAI*, 5363–5370.

Pensky, M.; Zhang, T.; et al. 2019. Spectral clustering in the dynamic stochastic block model. *Electronic Journal of Statistics* 13(1): 678–709.

Qin, T.; and Rohe, K. 2013. Regularized spectral clustering under the degree-corrected stochastic blockmodel. In *Advances in neural information processing systems*, 3120–3128.

Serrano, S.; and Smith, N. A. 2019. Is attention interpretable? *arXiv preprint arXiv:1906.03731* .

Stella, X. Y.; and Shi, J. 2003. Multiclass spectral clustering. In *Proceedings Ninth IEEE International Conference on Computer Vision*, 313–319. IEEE.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* .

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* .

Xu, D.; Cheng, W.; Luo, D.; Liu, X.; and Zhang, X. 2019a. Spatio-Temporal Attentive RNN for Node Classification in Temporal Attributed Graphs. In *IJCAI*, 3947–3953.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2019b. Self-attention with functional time representation learning. In *Advances in Neural Information Processing Systems*, 15915–15925.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive Representation Learning on Temporal Graphs. *arXiv preprint arXiv:2002.07962* .

Xu, K. 2015. Stochastic block transition models for dynamic networks. In *Artificial Intelligence and Statistics*, 1079–1087.

Xu, K. S.; Kliger, M.; and Hero, A. O. 2010. Evolutionary spectral clustering with adaptive forgetting factor. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2174–2177. IEEE.

Yang, T.; Chi, Y.; Zhu, S.; Gong, Y.; and Jin, R. 2011. Detecting communities and their evolutions in dynamic social networks—a Bayesian approach. *Machine learning* 82(2): 157–189.

Ying, Z.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in neural information processing systems*, 9244–9255.

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, 5165–5175.

Zhuang, D.; Chang, M. J.; and Li, M. 2019. DynaMo: Dynamic community detection by incrementally maximizing modularity. *IEEE Transactions on Knowledge and Data Engineering* .