Approximate Orthogonal Spectral Autoencoders for Community Analysis in Social Networks

Scott Wahl* and John Sheppard†, *Fellow, IEEE*
Gianforte School of Computing, Montana State University
Bozeman, Montana, USA
Email: *scott.wahl.us@gmail.com, †john.sheppard@montana.edu

***Abstract*—Discovery and partitioning of complex graphs such as social networks into communities can be useful for analyzing behavior of individuals in the network. Spectral clustering is one useful tool for performing this clustering, but it suffers from scalability and generalizability to new data. In this paper we introduce an approximate orthogonal spectral autoencoder. We apply this model to a political campaign contribution social network to show its effectiveness on out-of-sample embedding for clustering and classification. The resulting embedding is used with hierarchical fuzzy spectral clustering to show embedding generalization for a behavioral prediction problem for nodes in the social network.**

I. Introduction

Complex networks, or graphs, are a large and growing area of important research. Social networks are a type of graph that are generated from interactions among members of some population. These networks are used in many different fields such as genetics, neuroscience, collaboration networks, Internet groups, animal social behavior, and many more.

At their core, networks or graphs consist of points or *vertices* that are connected by *edges*. In many real-world networks, these edges are not distributed evenly throughout the graph, but instead there are groups of vertices that tend to have more connections among themselves than connections to the rest of the network [1]. These groups are commonly referred to as *communities*. Community detection is an important method for simplifying complex systems that can be difficult to analyze as a whole. Large scale graphs like social networks are one area where this analysis can provide interpretability, discover patterns in behavior, or find missing relationships.

Much of the early research in community analysis has focused on communities where vertices are assumed to belong to only a single community. This approach results in what is called a crisp community assignment [2, 3]. A popular algorithm for crisp community discovery is spectral clustering since it is relatively easy to implement and can find non-convex clusters [4, 5]. However, there is a limitation with restricting vertices in this way since for many social networks, it is possible for a vertex to belong to multiple communities.

Another limitation in some methods is that they do not account for sub-groups within a community. Such sub-groups can consist of smaller groups of individuals within a larger community, forming a hierarchy of communities. Military, business, familial, and political hierarchies are all examples of hierarchies where smaller groups combine to create a larger group. There are more recent approaches that attempt to improve on older algorithms by allowing fuzzy clusters as well as creating a hierarchical structure [6, 7, 8, 9].

In this paper, we examine approximate methods for finding encodings for social networks. We develop two mini-batch spectral embedding methods using an orthogonal spectral autoencoder (OSAE). The first applies an orthogonal constraint and loss on the encoding layer in addition to a reconstruction error. The second adds an approximate spectral decomposition by creating a smaller sampled Laplacian and performing eigenvector decomposition on that Laplacian. This adds a loss term for the difference between the orthonormal encoding and the top-k eigenvectors of the approximate Laplacian.

Fuzzy c-means clusters are used as community assignments for entities within the social network. For example purposes, we use federal campaign finance networks and voting history for the legislators within the graph. Each legislator is assigned to their communities based on the results from fuzzy c-means. Then those community assignments are used as an ideological approximation and combined with the voting data to generate a classifier for voting behavior. The results of these experiments show that the network embeddings are effective in predicting behavior of actors within the social network and are comparable in performance to computing a full hierarchical spectral decomposition of the graph.

II. Motivation

Based on data from the National Institute on Money in Politics[1], contributions to candidates and committees in 2016 reached $5.5 billion for federal elections and $3.70 billion for state level politics. For federal House and Senate candidates, donations reached roughly $1.7 billion. The majority of that money comes from a relatively small number of donors; there were only 11,479 donors who gave $10,000 or more. The amount from this group was over $804 million, nearly half the total amount. The remaining half came from roughly 708,000 entities. With such sums, the question of how such money may impact legislators and legislation is an important one.

It has been shown that a benefit to donating to a politician is that it provides access to that politician. While there is no clear evidence that standard political donations directly influence

[1]Based on numbers provided by https://www.followthemoney.org. Note that the first author is a former employee of this organization.

978-1-6654-6283-9/22/$31.00 ©2022 IEEE
DOI 10.1109/ICMLA55696.2022.00188

legislation, one experiment used a political organization attempting to arrange meetings with legislators and constituents who had previously donated and showed that those who donate can more easily get their legislators to listen to them [10]. Whether or not they revealed if the attendees had donated was decided randomly, allowing a better analysis on how donations impact access. From their results, only 2.4% of the offices arranged meetings when only told the attendees were constituents. However, 12.5% arranged meetings for attendees who were donors. Of those meetings, only 5.5% of the constituents met with a senior staffer, compared with 18.8% for donor attendees.

Another model shows how legislators may elect to adopt certain policy choices based on how interest groups donate [11]. Notably, even without the expectation of *quid pro quo*, this money can have an impact based on their temporal model. Since incumbents need to raise money for reelection, their knowledge of interest group donations can bias policy choices even in cases where the interest group has donated to the opposition. Some improvement can be made to these models since they do not fully capture the group behavior of donations.

III. Network Communities

We define a network, or graph, as $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ where $\mathbf{V}$ is a set of vertices, or nodes, in the graph.. $\mathbf{E}$ is a set of edges connecting the vertices of the graph. Edge $e_{ij} \in \mathbf{E}$ defines a connection between vertices $v_i, v_j \in \mathbf{V}$. For the purposes of this work, edges are considered to be undirected and some measure of affinity between vertices is associated with each edge. The degree d_i of vertex v_i is the sum of the edges connected to that vertex, i.e.

$$d_i = \sum_{j=1}^{n} e_{ij}.$$

Most real-world networks have been found to have structural properties that help inform or assist in learning useful knowledge concerning the entities within the network. Usually, such real-world graphs are not regularly structured like a lattice, i.e., the connections between vertices in the graph are not evenly distributed throughout the graph. Instead, there are substructures within the graph where vertices in that group have a higher proportion of edges connected to other members of that group than with members of other groups. These substructures are commonly called *communities*. We define a community $\mathbf{C}_i \subset \mathbf{V}$ as a subset of vertices. Using a simple definition of community structure, nodes within a community should have a high proportion of in-group edges compared with the rest of the network. Examples of networks that have community structure can be drawn from social [12], biological [13], gene expression [14], and many other types of networks [15]. Since the communities can represent fundamental properties of the network, their discovery is important for understanding the nature of the networks [1, 16].

IV. Graph-Based Clustering

Given the foundation covered in the previous section, our task becomes discovering communities within a graph. Note that there is already a wealth of research on finding communities. Some initial work focused on crisp partitioning of the network into non-overlapping, non-hierarchical communities [1, 3]. In this domain, a crisp partition is one where a vertex can only be a member of a single community. Thus, if $v_i \in \mathbf{C}_j$, then $v_i \notin \mathbf{C}_k, \forall \mathbf{C}_k : k \neq j$. A variety of approaches have been developed for finding communities in networks [2, 17], and one popular method is spectral clustering [4, 18]. These have proved popular for their ease of implementation, and for their ability to handle non-convex clusters.

To best represent the communities, organizing the vertices into clusters should satisfy two important realities of many social networks: cluster overlap and hierarchy. For the first, nodes within the network may belong to multiple communities. Much like in human social groups, an individual may belong to more than one community or have multiple affiliations, thus suggesting applying a fuzzy extension to community detection [19]. Hierarchy is another important component of some social networks wherein smaller communities can combine into larger ones. Military, business, and political hierarchies are all examples of hierarchies where individual smaller groups combine to create a larger group.

To find fuzzy communities, a variety of approaches have been presented. Palla uses a clique percolation method to find adjacent cliques with overlapping nodes [8]. Other methods use fuzzy modularity and simulated annealing or other techniques to find relevant partitions [7, 20, 21]. Fuzzy c-means is another possibility for determining fuzzy clusters and has been used to find hierarchies of clusters [6, 9]. The approach presented here differs in its use of spectral clustering and spectral characterization to create a top-down algorithm for finding hierarchical fuzzy clusters.

It is well known that K-means clustering does not handle non-convex clusters well. This is because the results of K-means clustering are biased towards spherical clusters centered on the found centroids. Spectral clustering is a popular alternative that can handle non-convex clusters due to its focus on graph topology [4]. Spectral graph partitioning methods already existed that relied on repeatedly cutting a network into smaller partitions [1], where it had been shown that the second eigenvector of a graph's Laplacian could be used to find an approximation of an optimal partition, typically by the sign of the entries in the vector. Each new partition would then be divided by isolating its nodes and performing the split again. Instead, spectral clustering uses the top k eigenvectors of the Laplacian instead of doing an iterative split [4].

The fuzzy spectral clustering algorithm starts by calculating the Laplacian matrix $\mathbf{L}_{sym}$. Then the top-k eigenvectors, correspond to the k smallest eigenvalues for the normalized symmetric Laplacian, are calculated from $\mathbf{L}_{sym}$ [4, 19, 22, 23]. These eigenvectors are oriented as the columns of matrix $\mathbf{X}$. After normalizing each row of $\mathbf{X}$, fuzzy c-means is used to

Algorithm 1 Fuzzy Spectral Clustering

function FSC($\mathbf{A}, k$)
  $\mathbf{D} \leftarrow \{d_i = \sum_1^n a_{ij}\}$
  $\mathbf{L}_{\text{sym}} = \mathbf{I}^{n \times n} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$
  $\mathbf{Z} = \text{eigs}(\mathbf{L}, k)$ ▷ Top-k eigenvectors.
  $\mathbf{X} = [\mathbf{z}_1, \ldots, \mathbf{z}_k]$
  for all row $\mathbf{x}_i \in \mathbf{X}$ **do**
    $x_{ij} = x_{ij} / \|\mathbf{x}_i\| \; \forall \, x_{ij} \in \mathbf{x}_i$ ▷ Normalize rows.
  end for
  $\mathbf{U} = \text{FCM}(\mathbf{X})$ ▷ Fuzzy c-means.
  return $\mathbf{U}$
end function

Algorithm 2 Hierarchical Generation

function HFSC($\mathbf{A}, k$)
  for $i = 2$ to k **do**
    $C_i = \text{FSC}(\mathbf{A}, i)$
  end for
  for all $C_{i,m} \in C_i : 3 \leq i < k$ **do**
    $P_{i,m,n} = \arg\max \mathcal{J}_f(C_{i,m}, C_{i-1,n})$
  end for
end function

find the cluster assignments $\mathbf{U}$ for each of the vertices.

To obtain hierarchical structure, the process is repeated by varying k corresponding to the number of clusters in each hierarchical level, as shown in Algorithm 2. In practice, calculating the eigenvectors can be performed once with the largest k and reused in subsequent iterations of clustering. The communities are calculated for each level using FSC to create a set of communities at hierarchical level i. Each community in level i is connected to its previous by calculating the fuzzy Jaccard similarity measure $\mathcal{J}_f$ of the communities. $\mathcal{J}_f(\mathbf{C}_i, \mathbf{C}_j)$ between two fuzzy communities $\mathbf{C}_i$ and $\mathbf{C}_j$:

$$\mathcal{J}_f(\mathbf{C}_i, \mathbf{C}_j) = \frac{\sum_{u_k \in \mathbf{C}_1 \cup \mathbf{C}_2} \min(C_{1,i}, C_{2,i})}{\max(C_{1,i}, C_{2,i})}.$$

The results give similarity measures for the smaller clusters that can be used to assign each cluster to its best matching parent. In practice, the possible parent of a child that has the highest similarity is selected as the parent of the community.

V. Political Networks

We construct the political social networks used in our experiments from campaign finance datasets. This data includes transactions where individuals, businesses, or other organizations make donations to political groups. These groups can be candidates, politicians, or political committees and form the vertices in the network. The donations to the political groups correspond to the edges in the network.

The dataset of political contributions is provided by Bonica and Stanford's Social Science Data Collection [24, 25]. As provided, the Stanford dataset uses two separate identifiers for candidates and donors. Using them separately would cause duplication of nodes in the network since an individual may both donate and receive money. To address this, the candidate (or recipient) data include the donor id so that it is possible to bridge the two separate ids and create a single unique identifier for an entity. This new identifier is used to populate the edges in the induced social network.

Recent work by Bonica [25] created an ideological estimate for both donors as well as candidates. In his work, contributions are assumed to represent evaluations of a candidate's ideology, and donors would be more likely to donate to those who share ideology. The resulting common-space campaign finance score (CFscore) has the advantage of applying to both types of entities, where prior research focused solely on legislators or recipients.

Calculating the CFScore for federal contribution data begins by creating an $n \times m$ contingency matrix $\mathbf{R}$. The rows of $\mathbf{R}$ map to contributors while the columns map to recipients. Each entry $r_{ij} \in \mathbf{R}$ contains a sum of the contributions from i to j. From this matrix, each entry is converted into an integer in the range $[1, 50]$ by dividing r_{ij} by 100 and capping the result to 50. This value is standardized further by dividing each r_{ij} by $\sum_i \sum_j r_{ij}$. From this matrix, singular value decomposition (SVD) is performed to obtain $\mathbf{K} = {\mathbf{D_r}}^{-\frac{1}{2}}(\mathbf{R} - \mathbf{r}\mathbf{c}^\top){\mathbf{D_c}}^{-\frac{1}{2}}$ where $\mathbf{r}$ and $\mathbf{c}$ are vectors of the row and column sums of $\mathbf{R}$. Additionally, $\mathbf{D_r}$ and $\mathbf{D_c}$ are diagonal matrices where the values of $\mathbf{r}$ and $\mathbf{c}$ are placed on the diagonal. From this, ideal points can then be estimated using $\theta = \mathbf{U}{\mathbf{D_c}}^{-\frac{1}{2}}$ for contributors and $\delta = \mathbf{V}{\mathbf{D_r}}^{-\frac{1}{2}}$ for recipients. Then $\mathbf{U}$ gives the left eigenvectors of $\mathbf{K}\mathbf{K}^\top$, and $\mathbf{D_r}$ is a diagonal matrix of singular values. Furthermore, $\mathbf{V}$ is made up of the right eigenvectors of $\mathbf{K}^\top\mathbf{K}$. For state data, these federal ideal points are used as bridge observations in an iterative procedure that estimates contributor and recipient CFScores across states.

Using the direct contributions, an initial network is created out of all the donations where the edge is weighted as $a_{ij} = amount$, and $amount$ is the sum of all donations or receipts between entities i and j. In addition, any node i with degree $d_i = 1$ is removed from the network since it lacks sufficient data to be associated with a community. The largest connected component of the remaining network is then used as the network of contributions. Removed nodes were few and represented smaller graphs of just a few entities isolated from all others. Thanks to the work by Bonica and Voteview, there is information tying the donations to federal legislators with their voting history dating back to 1980 [24, 26].

Using the communities discovered from the campaign finance dataset, we examined whether it was possible to generalize the community assignments to predict voting behavior in the legislature based on estimated ideological score of the bills themselves. To do so, we merged the community assignment features with the voting data and implemented random forest classifiers to predict Yea or Nay votes. As one way to see if donation amounts affect voting, multiple weighting schemes were tested for the connections between donors and recipients. In addition to just adjacency, we found communities based on weighted networks using a logarithmic scale as well as the

raw donation sums.

VI. Approximate Spectral Clustering

As mentioned, since spectral clustering on large datasets has high space and time complexity, there have been several approximation methods for spectral clustering developed. Nyström approximation is one such algorithm used to calculate the eigenvector decomposition [27], which has been adapted for use in spectral clustering [28].

In the fast spectral clustering algorithm proposed by Choromanska et al. (Algorithm 3), the process begins by selecting l columns sampled uniformly without replacement from the affinity matrix. Those sampled columns form the matrix $\hat{\mathbf{A}}$. In the algorithm, $\hat{\mathbf{A}} \leftarrow \mathbf{A}(:, \mathbf{L})$ is a sampling operation where : is the set of all elements and $\mathbf{L}$ are the indices of the sampled columns. Two diagonal matrices $\mathbf{D}$ and $\mathbf{\Delta}$ are created based on the row sums of $\hat{\mathbf{A}}$. An approximate Laplacian $\hat{\mathbf{C}}$ is calculated using the two diagonal matrices and the sampled columns. A sampling of the rows of $\hat{\mathbf{C}}$ using the indices of the sampled columns is used to create $\mathbf{W}$. Then the best r-rank approximation is taken of $\mathbf{W}$ (usually using SVD) to obtain approximate eigenvectors given by $\hat{\mathbf{U}}$. The remainder of the procedure is the same as standard spectral clustering where the column matrix containing the eigenvectors is row normalized followed by clustering the resulting matrix.

Similar work regarding Nyström approximations was done by Pourkamali-Anaraki [29]. In this work the author treats a sample of the data as *landmarks*. These landmarks are drawn from the exact eigenvector decomposition as calculated from the data. A linear transformation from the landmark set to the full set is applied to approximate the spectral embedding of the original data.

Our approach involves training an autoencoder to recover the approximate spectral decomposition. One relevant example for work with autoencoders is that of mini-batch spectral clustering [30]. As the authors note, calculating the Laplacian of a dataset as well as the associated spectrum can require $O(n^2)$ in storage for the Laplacian, and $O(n^3)$ time to calculate the spectral decomposition. The primary motivation is to calculate the spectrum of the Laplacian by finding the principal eigenvectors $\mathbf{Z}$ without the direct calculation. They note this can be reworked to optimize the following trace problem

$$\arg\min \left\{ \mathrm{Tr}\left(-\frac{1}{2}\mathbf{Z}^\top \mathbf{L}\mathbf{Z} \right) \right\} : \mathbf{Z}^\top \mathbf{Z} = \mathbf{I}.$$

Specifically, they note that the orthonormality constraint causes $\mathbf{W}$ to lie on a Stiefel manifold. The Riemannian gradient on this manifold is $\mathbf{H} = \left(\mathbf{I} - \mathbf{W}\mathbf{W}^\top\right)\mathbf{G}$. This allows some theoretical guarantees that a stochastic gradient optimization will converge. In the case of this work, the authors use the normalized Laplacian $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}} A \mathbf{D}^{-\frac{1}{2}}$. In this form, the top-$k$ eigenvectors are those from the largest k eigenvalues.

Traditionally, spectral clustering algorithms cluster over an entire dataset. However, there is an out-of-sample issue where new data requires a full recalculation instead of an easily performed transformation. A possible solution to this issue is to use a neural network to find a transformation $f(\mathbf{x})$ mapping a data point to a corresponding row of an eigenvector decomposition. Using the top-k eigenvectors, the goal is to find $f(\mathbf{x}) = \mathbf{z}_i$ where $\mathbf{z} \in \mathbb{R}^k$. Prior work has applied this idea with a stacked autoencoder [31]. This method is a two-step procedure wherein the initial step is to train the network to encode the graphs using the autoencoder. Once the autoencoder is trained over the input data, the decoder portion is removed and the encoder is fine-tuned to learn the mapping from the input to the eigenvectors.

Algorithm 3 Fast Spectral Clustering

function FastSpectralClustering($\mathbf{A}, k, l, r$)
  $\mathbf{L} \leftarrow$indices of l sampled columns
  $\hat{\mathbf{A}} \leftarrow \mathbf{A}(:, \mathbf{L})$
  $\mathbf{D} \in \mathbb{R}^{n \times n} : \mathbf{D}_{ij} = \delta[i = j]\, 1/\sqrt{\sum_{j=1}^{l} \hat{\mathbf{A}}_{ij}}$
  $\mathbf{\Delta} \in \mathbb{R}^{l \times l} : \mathbf{D}_{ij} = \delta[i = j]\, 1/\sqrt{\sum_{j=1}^{l} \hat{\mathbf{A}}_{ij}}$
  $\hat{\mathbf{C}} \leftarrow \hat{\mathbf{I}} - \sqrt{\frac{l}{n}}\mathbf{D} \times \hat{\mathbf{A}} \times \Delta$
  $\mathbf{W} \leftarrow \hat{\mathbf{C}}(\mathbf{L}, :)$
  $\mathbf{W}_r \leftarrow$ best r-rank approximation to $\mathbf{W}$
  $\tilde{\Sigma} = \frac{n}{l}\Sigma_{W_r}$ and $\tilde{\mathbf{U}} = \sqrt{\frac{l}{n}}\hat{\mathbf{C}}$
  $\mathbf{Y} = \forall i X_{ij} / \|\mathbf{X}_i\|$
  return $\mathbf{U} \leftarrow K$-means($\mathbf{Y}$)
end function

Deep embedded clustering attempts simultaneously to learn the set of cluster centers in the feature space along with the parameters that map the data points into the reduced feature space [32]. Xie *et al.*'s algorithm alternates between updating the network parameters and updating the cluster centers. Using a kernel derived from the Student's t-distribution to calculate similarity between the embedded points and the cluster centers, a Kullback-Leibler divergence loss function is used for a selected target distribution. Note that these descriptions only cover a small fraction of the work being performed in this field. For example, there are many different techniques using neural networks for clustering [33, 34] or improving the out of sample clustering performance using embedding [35, 36, 37, 38].

VII. Orthogonal Spectral Autoencoder

Our approach allows for out-of-sample clustering as well as limiting the size of the Laplacian needed to calculate the eigenvectors such that $l \ll n$. The primary framework for the orthogonal spectral autoencoder is a deep undercomplete autoencoder. In this architecture, there are multiple hidden layers between the input and output. The encoding layer is termed undercomplete since the dimension of this layer is smaller than the input layer. This structure can be useful in performing feature extraction and dimensionality reduction on the data. In general, this type of autoencoder uses a reconstruction loss based on the distance from values of the input and output, commonly mean squared error.

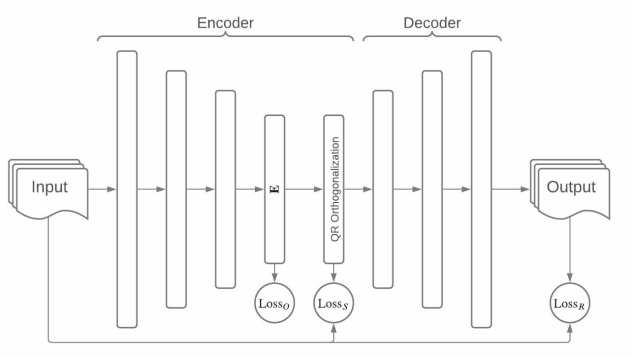

Figure 1: Structure of the Orthogonal Spectral Autoencoder

Figure 1 depicts the general structure of the orthogonal spectral autoencoder. Our additions to this framework include another layer between the encoder and decoder that orthogonalizes the output of the encoding layer $\mathbf{E}$ by using QR decomposition, where $\mathbf{E} = \mathbf{QR}$. There are two additional components to the loss function that penalize the encoding layers that are not orthogonal, in addition to approximating the eigenvectors of a sampled Laplacian.

Each of the hidden layers within the network are linear layers that apply the function $y = \mathbf{w}^\top \mathbf{H} + \beta$ where $\mathbf{w}$ is the weight vector, $\mathbf{H}$ is the input to the layer, and β is an additive bias. The output of each hidden layer uses the SELU activation function defined as

$$SELU\left(y\right) = s \cdot \left(\max\left(0, y\right) + \min\left(0, \alpha \cdot \left(\exp\left(y\right) - 1\right)\right)\right).$$

For our autoencoders, we use $k < b$, where k is the maximum number of clusters and the dimension of the encoding layer, and b is the size of the batch. Therefore, the matrix generated by the autoencoder at the final encoding layer is $\mathbf{E} \in \mathbb{R}^{b \times k}$. Then $\mathbf{Q}$ is a matrix such that the columns are orthogonal. Since $\mathbf{AR}^{-1} = \mathbf{Q}$, the inverse $\mathbf{R}^{-1}$ is saved within the QR Orthogonalization layer during each training step. The output of the QR Orthogonalization layer becomes $\mathbf{Z} = \mathbf{ER}^{-1}$. During evaluation, the last calculated $\mathbf{R}^{-1}$ is used to project the results of the encoding layer.

The loss function for OSAE comprises three parts. The first is the reconstruction error on the input data (Loss_R). In addition, there is a separate orthogonality loss (Loss_O), and a spectral loss (Loss_S). Total loss is then given by

$$\text{Loss} = \text{Loss}_R + \text{Loss}_O + \text{Loss}_S.$$

Reconstruction loss, Loss_R, is the mean squared error of the batch input $\mathbf{A}_b$ and the output of decoder portion of the autoencoder $\mathbf{A}'_b$,

$$\text{Loss}_R = \text{MSE}\left(\mathbf{A}_b, \mathbf{A}'_b\right) = \frac{1}{n}\sum_{i=1}^{n}\left(a_i - a'_i\right)^2$$

where $\mathbf{A}_b = \mathbf{A}\left[\mathcal{I}, :\right]$ and $\mathcal{I}$ are the indices of the minibatch input.

Loss_O is the orthogonalization loss and is defined as the mean absolute error of the identity matrix $\mathbf{I}^k$ and the product of the transpose of $\mathbf{E}$ with itself. By definition, if $\mathbf{E}$ is column orthogonal, then $\mathbf{E}^\top\mathbf{E} = \mathbf{I}^k$, so

$$\text{Loss}_o = \frac{1}{n^2}\sum_{i=1}^{k}\sum_{j=1}^{k}\left|\mathbf{I}^k_{ij} - \hat{e}_{ij}\right|$$

where $\hat{\mathbf{E}} = \mathbf{E}^\top\mathbf{E}$.

The spectral loss is calculated by using the orthogonal output of the QR decomposition layer and an estimate of the Laplacian $\mathbf{L}$. Since the input of the graph is a sparse adjacency matrix, slicing the matrix such that it contains only the rows and columns of the indices $\mathcal{I}$ of minibatch $\mathbf{A}_b$ would result in a mostly empty matrix. Instead, we use a pairwise Laplacian similarity measure

$$\mathcal{K}_L\left(\mathbf{A}_b, \mathbf{A}_b\right) = \exp\left(\frac{-\left\|a_{b,x} - a_{b,y}\right\|}{\sigma}\right)$$

where σ is the bandwidth for each pair of rows x and y in $\mathbf{A}_b$. In the experiments given later we use $\sigma = 2$. The pairwise Laplacian similarity $\mathcal{K}_L \in \mathbb{R}^{k \times k}$ yields a new affinity matrix $\mathbf{A}^L_b$ for minibatch $\mathbf{A}_b$. Using the pairwise Laplacian similarity, we calculate the normalized symmetric Laplacian of $\mathbf{A}^L_b$ as

$$\mathbf{L}_{sym} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}^L_b\mathbf{D}^{-\frac{1}{2}}$$

where $\mathbf{D}$ is a diagonal matrix, $d_{ii} = \sum_{j=1}^{k}\hat{a}_{ij}$, and $\hat{a}_{ij}$ is element i, j in $\mathbf{A}^L_b$. The next step calculates the k eigenvectors corresponding to the smallest eigenvalues of $\mathbf{L}_{sym}$ for $\mathbf{A}^L_b$. Matrix $\mathbf{Z}'$ is constructed by the top-k eigenvectors as given by the eigen decomposition $\text{eigs}\left(\mathbf{L}_{sym}, k\right)$. With this matrix, the spectral loss can be calculated as the mean absolute error of the approximate eigenvectors and the orthogonalized encoding,

$$\text{Loss}_S = \frac{1}{n}\sum_{i=1}^{b}\sum_{j=1}^{k}\left|\mathbf{Z}_{ij} - \mathbf{Z}'_{ij}\right|$$

For every training epoch, we sample b rows from the adjacency matrix $\mathbf{A}$. For a single batch, the rows of $\mathbf{A}$ are sampled uniformly without replacement to create minibatch $\mathbf{A}_b$. At the next epoch, a new batch is sampled in the same manner. We repeat this procedure until reaching the maximum number of training epochs. Future work may involve biasing the sampling to cover more training examples, or sampling without replacement to ensure no samples are repeated until all are covered.

The final trained network is then used to cluster the data. Using the orthogonal encoder portion of the network, the output of the QR decomposition layer ($\mathbf{Z}$) is used as the $n \times k$ embedding for the graph. Much like in spectral clustering, we normalize each row of $\mathbf{Z}$ as

$$\mathbf{z}^\ell_i = \frac{\mathbf{z}_i}{\sqrt{\sum_{j=1}^{k} z^2_{ij}}}.$$

The normalized rows of $\mathbf{Z}$ are used as the data points for fuzzy c-means clustering. The resulting $\mathbf{U}$ from FCM are

the fuzzy cluster assignments for the embedding. As before, it is possible to use the embedding for hierarchical spectral clustering as well. Since the spectral loss imposes an ordering on the columns of the output based on the ordering of the approximate spectral decomposition, we can use the ordered columns of $\mathbf{Z}$ similarly as the top-k' for clusters $k' < k$.

VIII. Prediction using OSAE Graph Embedding

We ran several experiments to test the efficacy of using orthogonal spectral autoencoders for vote prediction. The general procedure for performing the clustering is as follows. A weighted graph adjacency matrix is used as input to the neural network. Each element a_{ij} in $\mathbf{A}$ is scaled logarithmically by $a'_{ij} = \log\left(\sum a_{ij}\right)$ where a_{ij} is the value of a donation between nodes i and j in the graph.

For each year of data, we kept the structure and procedure of training the autoencoder the same. The only exception to this is that the input and output dimensions must change for each cycle to fit the different data. No other hyperparameters were changed between runs. All runs were done with a mini-batch size of $b = 128$. Based on the results from the clustering on the previous results, we set $k = 8$ as the maximum number of clusters, and thus the dimension of the encoding layer. Each network was trained over 500 epochs.

The effectiveness of the graph embedding for predicting future behavior was tested for each 2-year cycle separately. In each cycle, a random forest was trained to predict if a legislator would vote Yea or Nay based on their community assignments and the ideological estimate of the bill in question. A sample of $200,000$ votes was drawn from the relevant set of Yea and Nay votes for that two-year cycle. The random forest was constructed with an ensemble of 50 trees. As we are more interested in the relative performance, the random forests were not optimized or pruned.

The first experiment listed in the results is a baseline for comparison. Multiple trials were performed using the spectral decomposition with fuzzy hierarchical spectral clustering. In each trial, the spectral decomposition with $k = 8$ clusters calculated for each of the networks in the even (election) years $1980, 1982, \ldots, 2012$. For each of those years, we used three separate feature sets to predict behavior of the community members. The three separate feature sets are 1) the row normalized top-k eigenvectors of the normalized symmetric Laplacian $\mathbf{L}_{sym}$, 2) the hierarchical fuzzy c-means community assignments for each $k = 2, \ldots, 8$, and 3) the CFScore ideological estimates. This was done to add more context to the relative performance of the algorithms.

Each of the features was assigned to the individual nodes within the graph. These features were then attached to the voting history that pairs legislators within the graph to bills. Each bill has its own ideological estimate from DW-NOMINATE [39] that gives an estimate of the policy position of each bill. These values are generated by solving a utility model that attempts to maximize the obtained utility of voting Yea or Nay on a bill by a legislator. The closer a bill's ideology matches a legislator's ideology, the higher utility is obtained by voting Yea. The authors claim two dimensions for ideology worked well in practice. One dimension corresponded to an economic spectrum, and the other dimension to social issues.

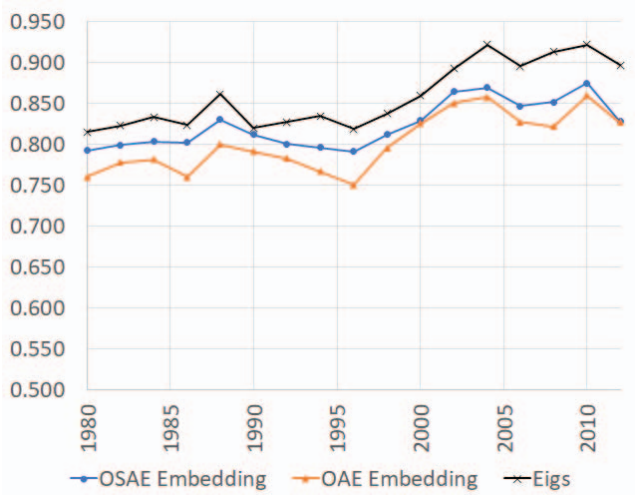

Figure 2: Vote Prediction using Spectral Embedding

The next experiments tested the efficacy of the embeddings for an orthogonal autoencoder and OSAE. Each network was trained for 500 epochs, where every epoch sampled a mini-batch from the adjacency matrix and used stochastic gradient descent to update the weights. The row normalized graph embedding $\mathbf{Z}$ was then used in the calculation of the features assigned to each legislator in the campaign finance network. As before, we analyzed the results of different treatments of the graph embedding: embedding itself, fuzzy c-means clustering, and hierarchical fuzzy spectral clustering. For all the following experiments, trials were repeated 10 times to get the average performance of each experiment. We analyzed the results of training the encoders for each cycle of the data.

The results of vote prediction on the spectral embedding of $\mathbf{Z}$ are shown in Figure 2. In general OSAE outperforms OAE in predicting behaviors among the legislators. The average performance per cycle of using the exact eigenvectors directly in prediction are also shown as a baseline.

Performing hierarchical fuzzy spectral clustering on the graph embeddings yields the results shown in Figure 3. The out-of-bag accuracy results for this classifier show it is not quite as effective as using the spectral embeddings themselves, though the results are still very close. This is similar to results from earlier experiments where HFSC was similar in effectiveness to using eigenvectors directly.

Figure 4 shows the results of the vote classification when using fuzzy c-means to cluster the spectral embedding at $k = 8$. As before, the embeddings of OSAE outperformed those of OAE, though not always to a significant degree. For 2010 where OAE had better out-of-bag accuracy in the random forest, it was not statistically significantly better than OSAE based on a Student's t-test. Still, this shows that the orthogonalization in OAE is useful in finding a network embedding

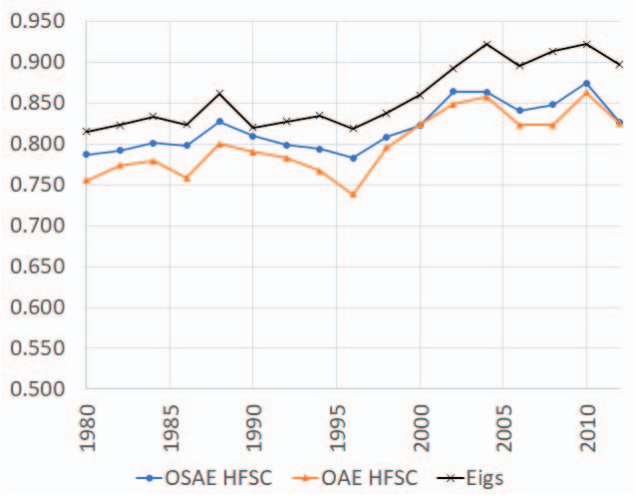

Figure 3: Vote Prediction using HFSC of Spectral Embedding

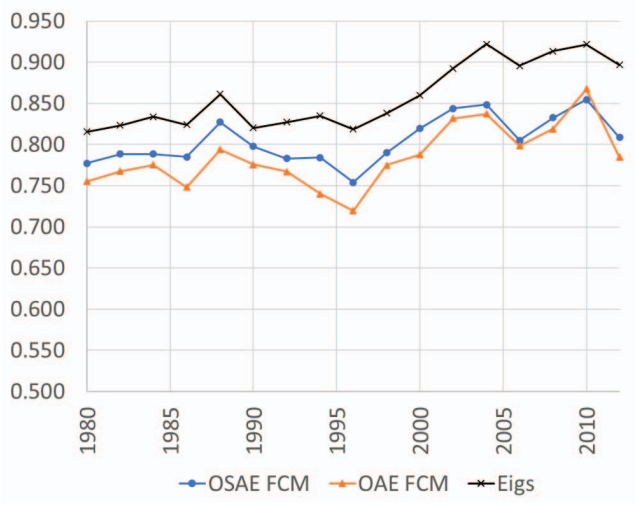

Figure 4: Vote Prediction using Fuzzy C-Means of Spectral Embedding

for predicting behavior of the communities. These results help highlight that predictions using HFSC are quite close to the performance of the spectral embeddings themselves and can outperform using FCM clustering.

IX. Conclusion

The orthogonal spectral autoencoder neural network allows embedding a graph into a lower dimensional space. Results show the orthogonal spectral loss function is effective at obtaining a graph embedding that is useful in predicting behavior of the individuals in the graph. The results of our experiments show performance comparable to the full, exact spectral decomposition. Based on these results, this avenue appears promising for refinement. There are many areas where improvements may be possible, either in modifying the structure of the autoencoder or additional hyperparameter tuning. Additional experiments will also be performed in the future to generalize the results to more datasets.

The introduction of the Orthogonal Spectral Autoencoder, in particular, opens many avenues for new research. As suggested, additional research in hyperparameter and structure tuning may yield a better approximation without further modifications to the underlying algorithm. Additions to the neural network architecture, such as graph convolutional networks, could tag the communities via embedding the node and edge heterogeneous information. These graph convolutional networks could also be used in a recurrent network to capture the dynamic properties of the social networks.

References

[1] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review*, vol. E 69, no. 026113, p. 026113, 2004.

[2] V. Blondel, J. Guillaume, R. Lambiotte, and E. Mech, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 10, p. P10008, 2008.

[3] M. E. J. Newman, "Modularity and community structure in networks," *Proc. National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[4] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*. MIT Press, 2001, pp. 849–856.

[5] A. Pothen, H. Simon, and K. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 3, pp. 430–452, 1990.

[6] A. Devillez, P. Billaudel, and G. V. Lecolier, "A fuzzy hybrid hierarchical clustering method with a new criterion able to find the optimal partition," *Fuzzy Sets and Systems*, vol. 128, no. 3, pp. 323 – 338, 2002.

[7] J. Liu, "Fuzzy modularity and fuzzy community structure in networks," *The European Physical Journal B*, vol. 77, no. 4, pp. 547–557, 2010.

[8] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[9] V. Torra, "Fuzzy c-means for fuzzy hierarchical clustering," in *Proc. IEEE Int. Conf. on Fuzzy Systems (FUZZ)*, May 2005, pp. 646–651.

[10] J. L. Kalla and D. E. Broockman, "Campaign contributions facilitate access to congressional officials: A randomized field experiment," *American Journal of Political Science*, vol. 60, no. 3, pp. 545–558, 2016.

[11] J. Fox and L. Rothenberg, "Influence without bribes: A noncontracting model of campaign giving and policy-making," *Political Analysis*, vol. 19, no. 3, pp. 325–341, 2011.

[12] S. Fortunato, "Community detection in graphs," *CoRR*, vol. abs/0906.0612, 2009.

[13] J. D. Power, A. L. Cohen, S. M. Nelson, G. S. Wig, K. A. Barnes, J. A. Church, A. C. Vogel, T. O. Laumann, F. M. Miezin, B. L. Schlaggar, and S. E. Petersen, "Functional network organization of the human brain," *Neuron*, vol. 72, no. 4, pp. 665–678, 2011.

[14] B. Zhang and S. Horvath, "A general framework for weighted gene Co-Expression network analysis," *Statistical Applications in Genetics and Molecular Biology*, vol. 4, no. 1, pp. Article 17+, 2005.

[15] M. Porter, J. Onnela, and P. Mucha, "Communities in networks," *Notices of the AMS*, vol. 56, no. 9, pp. 1082–1097, 2009.

[16] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee, "Self-organization and identification of web communities," *IEEE Computer*, vol. 35, pp. 66–71, 2002.

[17] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Computer and Information Sciences - ISCIS*, vol. 10, 2004, pp. 284–293.

[18] A. Pothen, H. Simon, and K. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 3, pp. 430–452, 1990.

[19] S. Zhang, R.-S. Wang, and X.-S. Zhang, "Identification of overlapping community structure in complex networks using fuzzy c-means clustering," *Physica A: Statistical Mechanics and its Applications*, vol. 374, no. 1, pp. 483–490, 2007.

[20] S. Bandyopadhyay, "Automatic determination of the number of fuzzy clusters using simulated annealing with variable representation," in *Foundations of Intelligent Systems*, ser. Lecture Notes in Computer Science, M.-S. Hacid, N. Murray, Z. Ras, and S. Tsumoto, Eds. Springer Berlin Heidelberg, 2005, vol. 3488, pp. 594–602.

[21] J. Xie, B. Szymanski, and X. Liu, "SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process," in *Proc. Int. Conf. on Data Mining Workshops (ICDMW)*, Dec 2011, pp. 344–349.

[22] S. Wahl and J. Sheppard, "Fuzzy spectral hierarchical communities in evolving political contribution networks," in *Proc. Int. Florida Artificial Intelligence Research Society Conf., FLAIRS*, 2017, pp. 371–376.

[23] ——, "Hierarchical fuzzy spectral clustering in social networks using spectral characterization," in *Proc. Int. Florida Artificial Intelligence Research Society Conf. (FLAIRS)*, 2015, pp. 305–310.

[24] A. Bonica, "Database on ideology, money in politics, and elections: Public version 1.0 [accessed 2013-6-27]," 2013. [Online]. Available: http://data.stanford.edu/dime

[25] ——, "Mapping the ideological marketplace," *American Journal of Political Science*, vol. 58, no. 2, pp. 367–386, 2014.

[26] J. B. Lewis, K. Poole, H. Rosenthal, A. Boche, A. Rudkin, and L. Sonnet, *Voteview: Congressional Roll-Call Votes Database [accessed 2018-04]*, Voteview, 2019, https://voteview.com/. [Online]. Available: https://voteview.com/

[27] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the nystrom method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 214–225, 2004.

[28] A. Choromanska, T. Jebara, H. Kim, M. Mohan, and C. Monteleoni, "Fast spectral clustering via the Nyström method," in *Algorithmic Learning Theory*, S. Jain, R. Munos, F. Stephan, and T. Zeugmann, Eds. Springer Berlin Heidelberg, 2013, pp. 367–381.

[29] F. Pourkamali-Anaraki, "Scalable spectral clustering with Nyström approximation: Practical and theoretical aspects," *IEEE Open Journal of Signal Processing*, vol. 1, pp. 242–256, 2020.

[30] Y. Han and M. Filippone, "Mini-batch spectral clustering," in *Proc. Int. Joint Conf. on Neural Networks (IJCNN)*, 2017, pp. 3888–3895.

[31] A. Jansen, G. Sell, and V. Lyzinski, "Scalable out-of-sample extension of graph embeddings using deep neural networks," *Pattern Recognition Letters*, vol. 94, pp. 1–6, 2017.

[32] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. on Machine Learning*, ser. ICML'16, vol. 48. JMLR, 2016, p. 478–487.

[33] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep spectral clustering using dual autoencoder network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4061–4070.

[34] M. T. Law, R. Urtasun, and R. S. Zemel, "Deep spectral clustering learning," in *Proc. Int. Conf. on Machine Learning*, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 1985–1994.

[35] F. Nie, Z. Zeng, I. W. Tsang, D. Xu, and C. Zhang, "Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering," *IEEE Transactions on Neural Networks*, vol. 22, no. 11, pp. 1796–1808, 2011.

[36] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet, "Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering," in *Proc. Int. Conf. on Neural Information Processing Systems*, ser. NIPS'03. Cambridge, MA, USA: MIT Press, 2003, p. 177–184.

[37] K. Levin, F. Roosta, M. Mahoney, and C. Priebe, "Out-of-sample extension of graph adjacency spectral embedding," in *Proc. Int. Conf. on Machine Learning*, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 2975–2984.

[38] C. Alzate and J. A. K. Suykens, "Multiway spectral clustering with out-of-sample extensions through weighted Kernel PCA," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 335–347, 2010.

[39] K. Poole, *Spatial Models of Parliamentary Voting*. Cambridge University Press, 01 2005.