A General and Scalable Approach to Mixed Membership Clustering

Frank Lin
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA
frank@cs.cmu.edu

William W. Cohen
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA, USA
wcohen@cs.cmu.edu

***Abstract*—Spectral clustering methods are elegant and effective graph-based node clustering methods, but they do not allow mixed membership clustering. We describe an approach that first transforms the data from a node-centric representation to an edge-centric one, and then use this representation to define a scalable and competitive mixed membership alternative to spectral clustering methods. Experimental results show the proposed approach improves substantially in mixed membership clustering tasks over node clustering methods.**

***Keywords*-clustering; scalable methods; unsupervised learning; large scale learning; mixed membership clustering;**

I. Introduction

Spectral clustering [1] is a data clustering paradigm where the bottom eigenvectors of a specific Laplacian (e.g., the Normalized Cuts Laplacian [2] or the symmetric normalized Laplacian [3]) of the affinity matrix of the data points are used to construct a low-dimensional embedding in which clusters are clearly separated in a metric space. Spectral clustering is popular due to its simplicity, effectiveness, and its ability to deal with non-linearly separable clusters.

There are two major drawbacks to spectral clustering methods. The first drawback is that they are computationally expensive because of the eigenvector computation, which is non-trivial even with faster sparse and approximate techniques [4]. The second drawback is that, as a type of *graph partition* method, unlike probabilistic topic models [5] or probabilistic network models [6], [7], spectral methods do not allow mixed membership clustering (overlapping clusters).

To address these problems, first we describe a method for converting a node clustering (graph partition) method into a mixed membership clustering approach by transforming a node-centric to a scalable edge-centric representation. Then we describe how this transformation can be used with *power iteration clustering* [8], a node clustering method based on spectral properties of the data (like spectral methods) that is experimentally competitive with spectral methods but more scalable. We show a substantial improvement in performance in mixed membership clustering tasks using the proposed approach.

II. Edges, Relationships, and Features

Instead of clustering the data instances, or the nodes in the graph, we want to cluster every feature occurrence, or edge, in the graph.

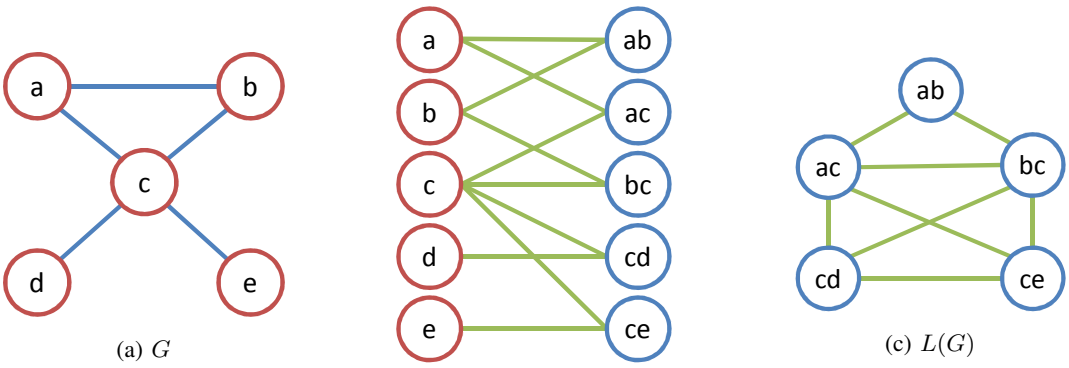
A central assumption we make in this work is that, while nodes in a graph can belong to more than one cluster, an edge between two nodes, indicating an affinity relationship, belongs only to one cluster. If we can determine the membership of these edges correctly, then we can assign multiple labels to the nodes based on the membership of the incident edges. In the context of a social network, edge clustering can be interpreted as relationship clustering—instead of forcing every person to belong to only one social community, each of the relationships will be assigned to a social community. For example, person a's relationships with a's parents, siblings, and cousins belong to the community of a's family and relatives, and a's relationships with his co-workers belong to the community of the company a works for. One-community-per-relationship is a much better assumption than one-community-per-person because it better fits our understanding of a social network structure and allows multiple community labels per person—in fact, person a can have as many labels as the number of relationships a has.

If we are to apply graph partition methods to edges of the graph, first we need to transform the graph so to an "edge-centric" representation. In this work, we will construct what we call a *bipartite feature graph* (BFG). A BFG $B(G)$ on a graph G is a bipartite graph satisfying the following conditions:

- $B(G) = (V_V, V_E, E)$ where V_V and V_E are disjoint sets of nodes and E is the set of edges.
- Each node in V_V corresponds to a node in G and each node in V_E corresponds to an edge in G.
- An edge $e \in E$ exists between nodes $a \in V_V$ and $b \in V_E$ if and only if node a is incident to edge b in G.

In other words, for each edge $e_i(u, v)$ in G, we add a new node e_i to the node set of $B(G)$, and connect e_i to the nodes u and v in $B(G)$.

We compare BFG to *line graphs*, a more common edge-centric representation. A line graph $L(G)$ on a undirected,



(a) G (b) $B(G)$ (c) $L(G)$

Figure 1: An example graph G and its corresponding bipartite feature graph $B(G)$ and line graph $L(G)$. In (b) and (c), the blue nodes represent the edges in G; e.g., edge ab represents the edge connecting a and b in (a).

unweighted graph G is a graph where (1) each node of $L(G)$ represents an edge of G, and (2) an edge exists between two nodes of $L(G)$ if and only if their corresponding edges in G have an incident node in common; In other words, for each edge $e_i(u, v)$ in G, we create a node e_i in $B(G)$, and connect e_i to e_j if u or v is also an end point of e_j.

There are several advantages to BFG as an edge-centric representation over line graphs; (a) unlike a line graph, the original graph G can always be constructed from a BFG $B(G)$, (b) $B(G)$ has the same space complexity as that of G, and (c) it is trivial to modify the BFG to correspond to a directed, weighted graph. The biggest drawback with line graphs is that they not scalable. For example, a set of nodes and edges in G that form a "star" pattern—one node in the middle connected to n nodes via n edges—translates to n nodes and n^2 edges in $L(G)$. This is especially a problem in most large, social networks that display a power-law distribution in the number of incident edges per node [9], [10].

A graph G can be represented as a square matrix A where the rows and columns correspond to nodes and a non-zero element $A(i, j)$ corresponds to an edge between nodes i and j; similarly, a BFG $B(G) = (V_V, V_E, E)$ can be represented as a rectangular matrix F where the rows correspond to V_E and the columns correspond to V_V, and a non-zero element $F(i, j)$ correspond to an edge between i and j, which in turn represents an incidence between edge i and node j in G. An immediate corollary is that F will always be sparse, since every row in F will always have only two non-zero elements, and the number of non-zero elements in F is $2m$ where m is the number of edges in G. Thus, while a BFG transformation does not increase the space complexity of the original input, methods that work with the BFG need to be sparse matrix-friendly in order to scale to large datasets.

An important observation we want to make here is that a $B(G)$ is a valid edge-centric representation of G regardless of G's structure, even if, let's say, G is itself a bipartite graph. This observation generalizes BFG to represent not just graphs, but any data represented by weighted feature vectors. Actually, a bipartite graph has been often used to represent large feature vector-based datasets such as noun phrases found on the web [11], [12], large document collections [13], [12], and social network communities [14], [15] for scalable semi-supervised learning and clustering analysis.

A bipartite graph can be constructed from a dataset of feature vectors X as follows. Let $\mathbf{x}_i \in X$ be the i-th instance of X, and let $\mathbf{x}_i(j)$ be the weight of $\mathbf{x}_i$'s j-th feature. Then create a bipartite graph $G = (V_I, V_F, E)$ where the i-th node in V_I correspond to the $\mathbf{x}_i$ and the j-th node in V_F correspond to the j-th feature, and an edge $e(i, j) \in E$ is weighted by $\mathbf{x}_i(j)$. The BFG can be applied as above and therefore we can transform any graph or dataset of feature vectors into an edge-centric representation.

An intuitive interpretation of edge clustering on the bipartite instance feature graph is that, instead of clustering instances, where each instance can belong to multiple clusters, we want to cluster each *feature occurrence*. For example, a text document on the subject of sports management may belong to both "sports" and "business administration" categories. However, we can assign each word occurrence to a specific category (e.g., the word "football" to the sports category and the word "budget" to the business category), and then assign the document multiple labels depending on the labels of its word features.

III. Edge Clustering

After transforming a graph into a BFG, edge clustering can be done with any graph-based clustering method, like spectral clustering methods such as Normalized Cuts [2] and the Ng-Jordan-Weiss method [3]. Then the cluster labels assigned to the edge nodes V_E in the BFG can be used to determine mixed membership labels for the nodes in the

original graph. Algorithm 1 outlines the steps for the general procedure, where A is the matrix representation of the input graph, k is the number of desired clusters, and Cluster and Labeler are the specified clustering method and the node label assignment strategy, respectively.

Algorithm 1 Proposed general mixed membership clustering method via edge clustering

1: **procedure** MMCLUSTER(A, k,Cluster,Labeler)
2: Transform A into a BFG and get $B(A)$
3: Run Cluster($B(A), k$) and get edge clusters $E_1, E_2, ..., E_k$.
4: Run Labeler($E_1, E_2, ..., E_k$) and get mixed membership node clusters $C_1, C_2, ..., C_k$
5: **return** $C_1, C_2, ..., C_k$
6: **end procedure**

An issue to consider when choosing the graph clustering method is its scalability. If the original graph $G = (V, E)$ is represented by a $|V| \times |V|$ matrix, then its BFG matrix is $(|E| + |V|) \times (|E| + |V|)$, a much larger matrix for most types of data. In order for this approach to scale to large datasets, the graph-based method must be able to take full advantage of the sparsity of BFG.

A. Power Iteration Clustering

For the graph-based clustering method we use *power iteration clustering* (PIC) [8]. In essence, PIC finds a very low-dimensional data embedding using truncated power iteration on a normalized pair-wise similarity matrix of the data points, and this embedding turns out to be an effective cluster indicator.

PIC is related to a family of clustering methods called *spectral clustering*. PIC and spectral clustering are similar in that both embed data points in a low-dimensional subspace derived from the affinity matrix, and this embedding provides clustering results directly or through a k-means algorithm. They are different in what this embedding is and how it is derived. In spectral clustering the embedding is formed by the bottom eigenvectors of the Laplacian of an affinity matrix. In PIC the embedding is an approximation to a *eigenvalue-weighted linear combination* of *all* the eigenvectors of the row-normalized affinity matrix. This embedding turns out to be very effective for clustering, and in comparison to spectral clustering, the cost (in space and time) of explicitly calculating eigenvectors is replaced by that of a small number of matrix-vector multiplications. Here we reproduce the original PIC algorithm as Algorithm 2.

The diagonal degree matrix D is defined as $D(i, i) = \sum_j A(i, j)$. Typically, the indicator $\mathbf{v}^0$ is first assigned uniformly random values, as in the power method for determining the principle component of a square matrix. The normalization in Step 5 can be used to keep the numerical values in $\mathbf{v}$ from overflow or underflow, and can also used to keep $\mathbf{v}$ a probability distribution (e.g., $\mathbf{v}^{t+1} \leftarrow \frac{\mathbf{v}^{t+1}}{\|\mathbf{v}^{t+1}\|_1}$).

Algorithm 2 The original PIC algorithm

1: **procedure** PIC(A, k)
2: Initialize $\mathbf{v}^0$ and $t \leftarrow 0$
3: **repeat**
4: $\mathbf{v}^{t+1} \leftarrow D^{-1}A\mathbf{v}^t$
5: Normalize $\mathbf{v}^{t+1}$
6: $\boldsymbol{\delta}^{t+1} \leftarrow |\mathbf{v}^{t+1} - \mathbf{v}^t|$ and $t \leftarrow t + 1$
7: **until** $|\boldsymbol{\delta}^t - \boldsymbol{\delta}^{t-1}| \simeq \mathbf{0}$
8: Use k-means on $\mathbf{v}^t$ to get clusters $C_1, C_2, ..., C_k$.
9: **return** $C_1, C_2, ..., C_k$
10: **end procedure**

B. Edge Clustering with PIC

The input to PIC is A, a non-negative square affinity matrix where $A(i, j)$ represents the similarity between instances i and j (or, in the context of a graph, the weight of the edge between node i and j). If we want to cluster the nodes of a graph G (assuming homophily), then we can simply use G for A. However, here we want to cluster the edges of G. As mentioned in Section II, the most direct edge-centric representation of G is the line graph $L(G)$, but the problem with $L(G)$ is that not only is it potentially a very dense graph, it will most likely be dense for many types of data such as social networks and document collections. The bipartite feature graph $B(G)$ is a more scalable representation, but it introduces another problem. A BFG, which is a bipartite graph, is always *periodic*, and therefore iterative algorithms such as the power iteration and PageRank [16] do not converge. Likewise, PIC, which is based on the power iteration, cannot be used on $B(G)$.

Here we propose to turn $B(G)$ into a unipartite graph as follows. Let $cn(i, j) \subseteq V_V$ be the set of common incident nodes of edge i and j in G, and let F be the matrix representation of $B(G)$ (as in Section II), and we define a similarity function $s(i, j)$ where $i, j \in V_E$ as follows:

$$s(i, j) = \sum_h \frac{1}{\sum_j F(j, h)} F(i, h) \cdot F(j, h) \qquad (1)$$

In other words, the similarity between edge i and j in G is the number of common incident nodes they have (at most 2), weighted proportionally to the product of the edge weights but inversely proportional to the number of edges each node is incident to. This is an intuitive similarity function between two edges that incorporates the number of common incident nodes, their weights, and the inverse frequency of the incident nodes[1]. Then we can define a square matrix S where $S(i, j) = s(i, j)$, and use S in place of A in Algorithm 2.

One final difficulty remains: S could still be dense. As

[1]The assumption is that a node should not be considered important for determining similarity if it is incident to many edges; e.g., two links both pointing to the popular search engine Google.com is hardly a evidence of their similarity. This is similar to the *tf-idf* term weighting commonly used for comparing document similarity in information retrieval methods [17]

Equation 1 implies, $s(i,j)$ is non-zero as long as i and j have one incident node in common—which brings us back to the problem with line graphs mentioned in Section II. However, a simple and efficient "trick" can be applied to obtain a solution with time and space complexity linear to the size of the input, based on a couple of observations:

1) The (likely) dense similarity matrix S can be decomposed as $S = FNF^T$, a product of sparse matrices, where the diagonal matrix N defined as $N(j,j) = \sum_i F(i,j)$.
2) A is only used in Step 4 of Algorithm 2 for a matrix-vector multiplication.

The above observation allows us to replace Step 4 of Algorithm 2 with the following:

$$\mathbf{v}^{t+1} \leftarrow D^{-1}(F(N(F^T\mathbf{v}^t))) \tag{2}$$

and produce the exact same result as using S directly. Note that the parentheses specifying the order of operations is important to keep all computations sparse matrix-vector multiplications linear to the input size. One more thing; since S is never explicitly constructed, D also cannot be computed directly. However, one can verify that we can compute it efficiently by computing a vector $\mathbf{d} = FNF^T\mathbf{1}$ (where $\mathbf{1}$ is a vector of 1's) and let $D(i,i) = \mathbf{d}(i)$.

C. Assigning Node Labels

After obtaining a cluster label for every edge, we can proceed to assign labels for every node based on the labels for its incident edges. Let $L(i,j)$ be the number of node i's incident edges assigned the j-th label. We propose three simple variations:

- **Max** The label assigned to node i is $argmax_j L(i,j)$. This will assign only one label to a node as in typical node-based clustering methods, and is useful for comparing against them in a single-membership setting.
- **T@p** Label j is assigned to node i if $\frac{L(i,j)}{\sum_j L(i,j)} \geq \frac{p}{100}$; i.e., T@20 means that node i will be assigned label j if at least 20% of its incident edges are assigned j. It falls back to Max if no labels meet the criteria.
- **All** Label j is assigned to node i if $L(i,j) \geq 1$; i.e., node i will be assigned all the labels of its incident edges.

Putting together the various parts of the method, the mixed membership clustering via edge-clustering using PIC, which call PIC_E, is outlined in Algorithm 3. Note that unlike Algorithm 1, PIC_E uses the compact representation F instead of $B(A)$.

Algorithm 3 Mixed membership clustering using PIC

1: **procedure** $\text{PIC}_\text{E}(A, k, p)$
2: Transform A into a BFG as F
3: Run PIC(F, k), replacing Step 4 with Equation 2, and get edge clusters $E_1, E_2, ..., E_k$.
4: Use T@p on $E_1, E_2, ..., E_k$ to get node clusters $C_1, C_2, ..., C_k$.
5: **return** $C_1, C_2, ..., C_k$
6: **end procedure**

IV. Experiments

For experiments we compare the proposed method with different node label assignment methods on a number of datasets. In addition we will use the original node-centric PIC and the closely related normalized cuts method (NCut) [2] as baselines. Instead of evaluating clustering methods indirectly using a supervised learning task as done in some previous work [14], [15], we want to compare output clusters directly with human assigned categories. For the evaluation metric we will report the macro-averaged F1 score[2], often used for multi-label categorization tasks [18], [17], after aligning output clusters with ground-truth category labels using the Hungarian algorithm [19]. We prefer this metric over label accuracy because the latter tend to inflate prediction performance when the cluster sizes are not balanced.

A. Modified Network Datasets

We gather a set of eight benchmark network datasets with single-membership ground-truth labels for our first set of experiments. For each dataset we synthesize mixed membership instances by randomly drawing and merging pairs of nodes. We prefer modifying a variety of existing real datasets over purely synthetic ones such as the *planted partition model* [20] as they may be a better predictor on how well these methods would perform on these types of real mixed membership datasets. Our primary goals for these experiments are:

- To verify that edge clustering works as well as node clustering on single-membership data.
- To vary the degree of "mixed membership-ness" and see how well each method does.

The merging process is follows. (1) Split nodes of graph $G = (V, E)$ into two sets S and T such that $S \cup T = V$ and $S \cap T = \emptyset$, such that $\frac{|S| \cdot 100}{|V|} \approx m$. (2) For each node $s \in S$, randomly select a node in $t \in T$, and add all labels and edges of s to t (i.e., add a new edge (t, c) if $(s, c) \in E$). (3) Remove all nodes in S and their incident edges.

The parameter m controls the degree of mixed membership-ness and goes from 0 to 100, where 0 would result in the original single membership graph and 100 would result in a graph with a single node with all possible membership labels. In between, m guarantees that at least $m\%$ of the nodes in original single membership graph G is merged in the mixed membership graph G'.

For the experiments this process is repeated 50 times per dataset and the reported evaluations are averaged over these

[2]The harmonic mean of precision and recall

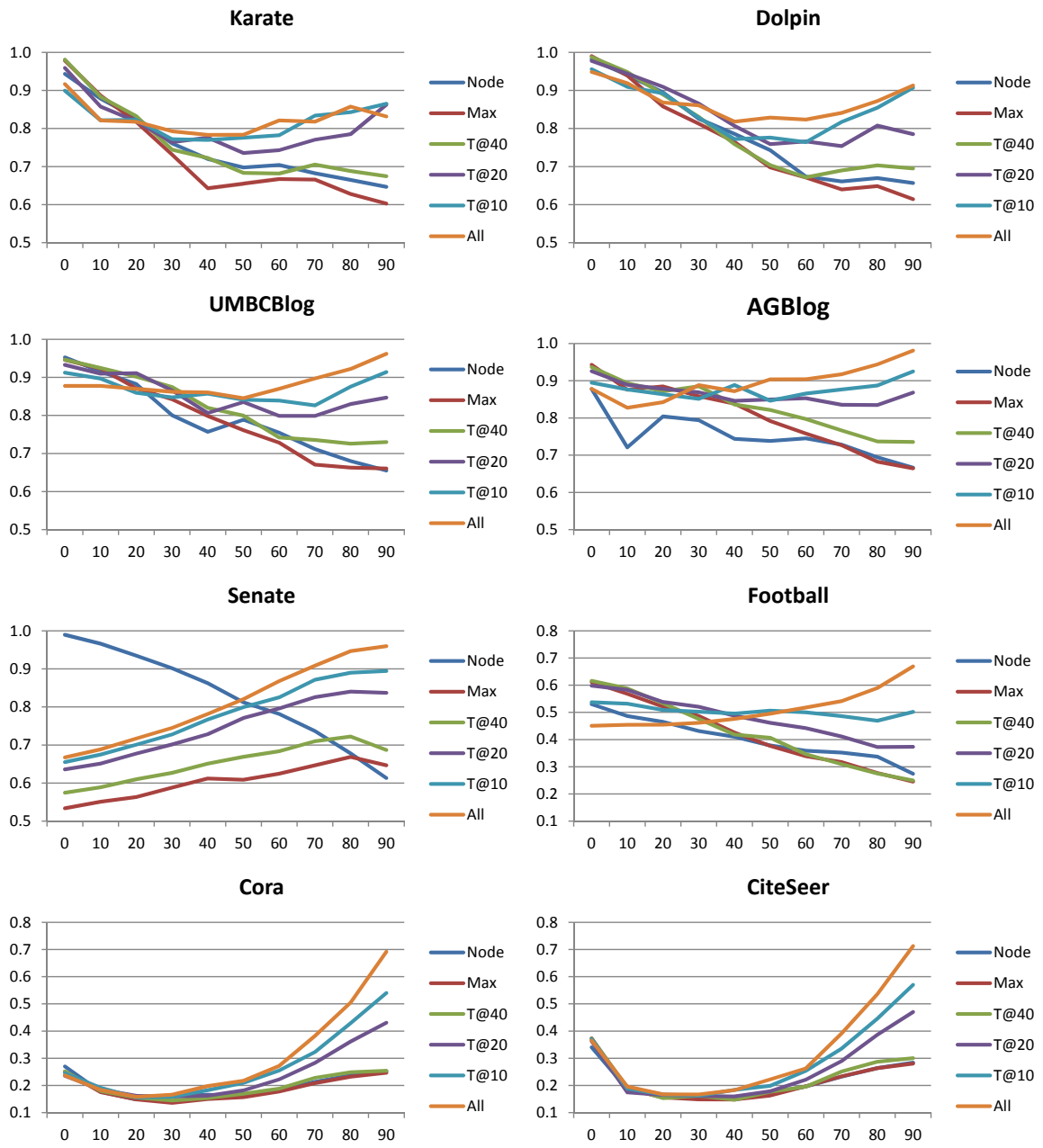


Figure 2: Mixed membership parameter m (x-axis) versus macro-averaged F1 score (y-axis) on the modified network datasets.

50 runs.

Here we briefly describe the eight datasets. The Karate dataset [21] form a two-community social network among 28 members a karate club; the nodes are people and the edges are friendships. The Dolphin dataset [22] is a two-community social network of associations between dolphins in a pod in New Zealand. The UMBCBlog [23] and AGBlog [24], datasets are networks of 404 and 1222 political blogs, respectively. The nodes are blogs and edges are hyperlinks between them, and each blog belong either to the *liberal* or *conservative* community. The Senate dataset contains nodes corresponding to 98 US senators and edges are agreement on congressional votes; labels correspond to affiliations with either a *liberal* or a *conservative* political party; unlike other datasets, this dataset is a complete graph. The nodes in the Football dataset [25] are 115 US Division IA colleges, each belonging to one of 10 conferences, and the edges represent games in the 2000 regular season. The Cora and CiteSeer datasets [26] are 2485 and 2114 scientific papers belonging to 7 and 6 related scientific fields, respectively; edges are

citations. All of these datasets have weighted, undirected edges.

The experiment results on the network datasets are shown in Figure 2, where Node refers to the original PIC algorithm using Node-based clustering. Results for NCut is nearly the same as that of Node and is not shown in the figure for sake of clarity. Here we make a few observations: (a) Except for Senate, on most datasets, edge clustering methods do just as well as Node for $m = 0$ (single membership). (b) As m increases, All and methods with a low p perform better as expected. (c) The performance of Max is very similar to that of Node—it does well at low m's but not at higher m's, whereas All usually is the worst (not by much) at low m's and best at high m's. (d) The poor performance of edge clustering methods on Senate suggests that they may not be well-suited for certain dense network datasets. (e) The threshold parameter p should be tuned for an optimal result—Max and All do not do well at particular extremes of m, while T@20 consistently outperform Node at almost any m, except on the Senate dataset. The results verify that edge clustering will generally work well on single-membership datasets as well as mixed membership ones. Note that at the very high end of p edge clustering methods, especially All, would do well simply because most instances will have membership in most classes.

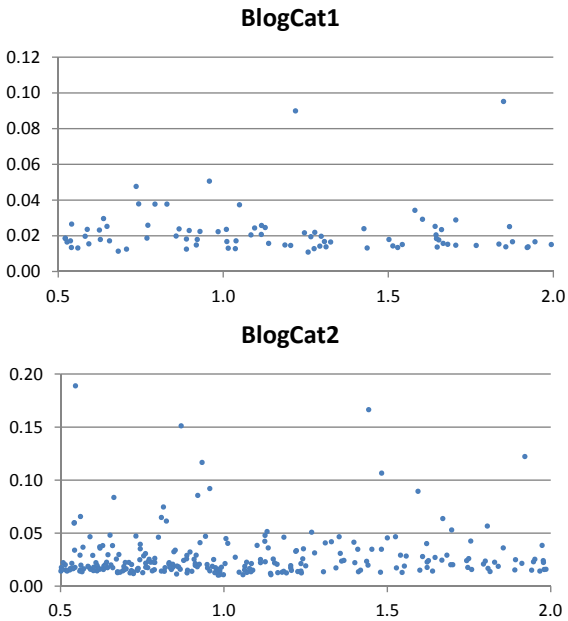


Figure 3: Summary statistics for the BlogCatalog datasets. Each dot on the chart is a category-pair dataset. The x-axes correspond to the size ratio between the two categories, and the y-axes correspond to the ratio of instances that belong to both categories.

B. BlogCatalog Datasets

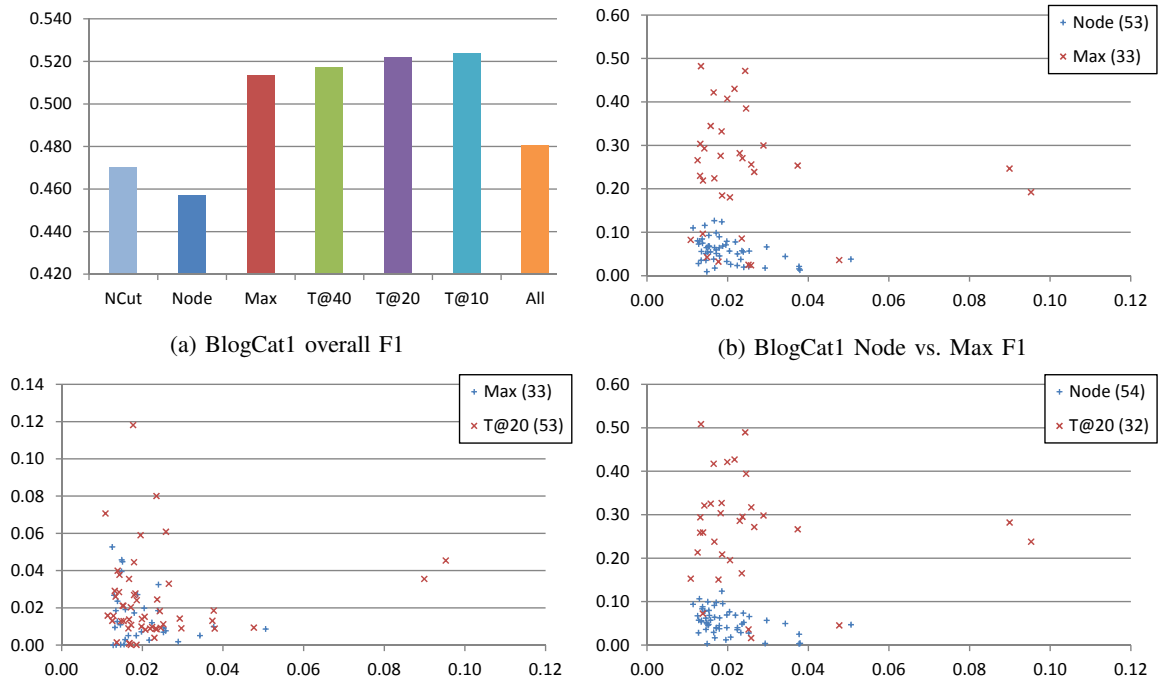
BlogCat1 and BlogCat2 [14], [15] are two blog datasets crawled from BlogCatalog.com, during two different time periods. BlogCat1 contains 10,312 blogs/users, links between these blogs, and each of the blog is manually assigned one or more labels from a set of 39 category labels. BlogCat2 is a similar dataset with 88,784 blogs and 60 category labels, and additionally each blog may be associated with a subset of 5,413 tags. For our experiments we will consider links between blogs and associated tags as input, and use the manually assigned category labels as gold-standard for evaluation.

Instead of evaluating a clustering method indirectly using a supervised learning task as done previously on these blog datasets [14], [15], we want to directly compare output clusters directly with human assigned categories. However, on a large, noisy dataset with many possible multi-category assignments, it may not be fruitful to compare all clustering and category assignments at once—many of the possibilities can be considered correct and the manually assigned categories may be deficient. Here we will tease out some of these kinds of noise by evaluating a clustering method on one *pair* of categories at a time, instead of the entire dataset.

We want to focus on cases where there are actual mixed membership instances, so we select category pairs where when instances belonging to them are pooled together, 2% to 70% of them are mixed membership (belong to both categories). The 70% cap is so there is enough signal there from either category to "guide" the clustering method in separating the data according to the selected categories. We also filter category pairs based on the size ratio between the two categories so that the number of instances in the larger of the pair is at most twice that of the smaller one. We end up with 86 category-pair datasets for BlogCat1 and 158 for BlogCat2. Figure 3 plots the pair size ratio against the ratio of mixed membership instances for these datasets. Note that most category pairs have only a small percentage of mixed membership instances. For each category-pair dataset, the same method is run 10 times (since all methods involve some random initialization) and the average F1 score of runs are reported.

For BlogCat1, we simply use the blog network as input. The overall F1 scores averaged over 86 datasets are shown in Figure 4a. An interesting observation here is that not only does edge clustering in general do better than node clustering, even Max is able to outperform both NCut and Node, which suggests that edge clustering may be better than node clustering even on single membership clustering tasks.

Figures 4b–4d are detailed comparison between two specified methods that require some explanation. Each marker on the plot correspond to one category-pair dataset, and the color and shape of the marker shows which method outperforms the other method, according to the legend in the upper right corner. The legend additionally shows the number of



(a) BlogCat1 overall F1

(b) BlogCat1 Node vs. Max F1

(c) BlogCat1 Max vs. T@20 F1

(d) BlogCat1 Node vs. T@20 F1

Figure 4: BlogCat1 dataset results. (a) is the overall F1 score averaged over all category pairs, and (b)–(d) are detailed per-category pair charts for detailed comparison of two specific methods.

times the method outperforms the other in parentheses. The x-axes correspond to the ratio of instances in the category pair that have membership to both categories, and the y-axes correspond to the margin by which the winning method outperforms the other on the category pair. For example, in Figure 4b, we can see that although Node outperforms Max on more category pairs (53), when Max outperforms Node it is often by a large margin, versus where the winning margins of Node are quite low, indicating that there is very little utility in choosing Node over Max. A general observation of Figures 4b–4d is that PIC_E with an appropriate labeling threshold is almost always better than single membership methods not just in terms of overall performance, but even on a per-dataset scale, and especially for datasets with a higher mixed membership ratio.

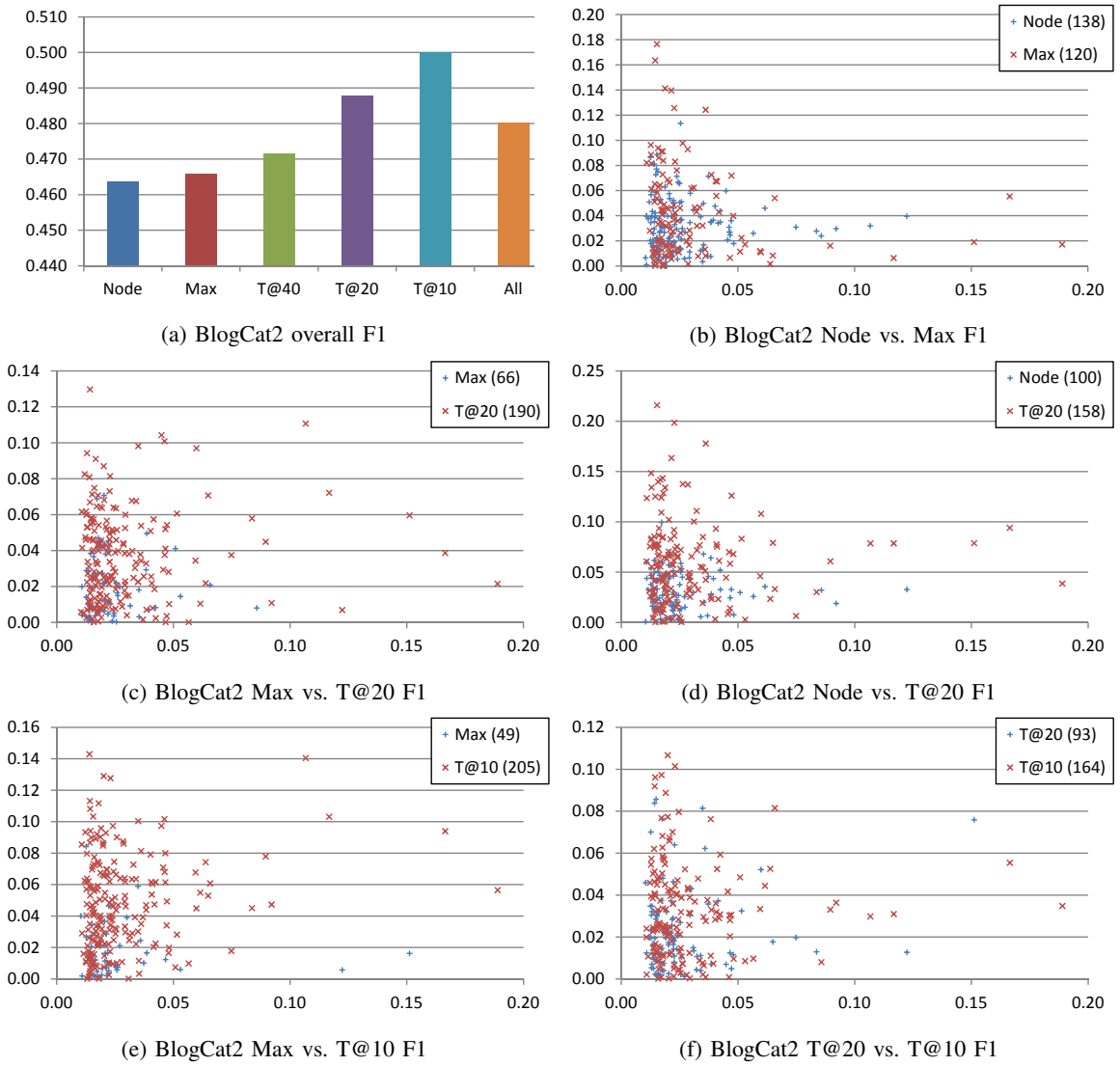
For BlogCat2, we want to take advantage of the additional tag information. To test PIC_E's performance on general features (not just networks) and even a mixture of different features, the BlogCat2 dataset input contains both links between blogs and tags associated with each blog. Each instance is a feature vector of both tags and links for each blog, which can be interpreted as a bipartite graph and transformed into a BFG as described in Section II. The results for BlogCat2 are shown in Figure 5, using the same types of plots as BlogCat1. Note that there are no NCut results for BlogCat2 since NCut does not take general feature vectors as input.

Unlike BlogCat1, where the overall edge clustering methods in Figure 4a are rather "flat" with respect to the label assignment threshold parameter p (except for All), the overall result in Figure 5a shows a trend for a specific node label assignment parameter p. This trend can be further examined in the detailed comparisons in Figures 5b–5f, showing that for most category pairs (a) edge clustering method outperform node clustering methods, (b) the methods with high "win" margins are edge clustering methods, and (c) edge clustering methods do better on category pairs with higher mixed membership ratios. In addition, BlogCat2 shows that for certain datasets tuning p is important for an edge clustering approach to output cluster labels that match the ground-truth categories.

V. Related Work

Palla et al. [27] emphasized the importance of recognizing *overlapping communities* in naturally occurring network data instead of just disjoint communities, and proposed a community discovery method that uses cliques in the graph as the basic structure for inferring communities. The time complexity of the method is exponential in the number of edges in the graph; therefore, even with a small exponent, it does not scale to large datasets.

Mixed membership stochastic blockmodels [6] are a probabilistic method that models both the pairwise presence of



(a) BlogCat2 overall F1

(b) BlogCat2 Node vs. Max F1

(c) BlogCat2 Max vs. T@20 F1

(d) BlogCat2 Node vs. T@20 F1

(e) BlogCat2 Max vs. T@10 F1

(f) BlogCat2 T@20 vs. T@10 F1

Figure 5: BlogCat2 dataset results. (a) is the overall F1 score averaged over all category pairs, and (b)–(f) are detailed per-category pair charts for detailed comparison of two specific methods.

links between objects in a network and a global "block" structure that indicates the interaction between clusters. This method provides a probabilistic framework and has the ability to learn parameters (e.g., the interaction between two clusters) and generate random networks based on these parameters. However, since each of the n^2 possible edges is a random variable in these models, they are in general not scalable to larger datasets.

EdgeCluster [14] is method for finding the *social dimensions* of a social network graph. Similar to our proposed method, it first constructs an edge-centric graph from the social network, where edges become nodes and nodes become edges. Then a modified version of k-means, which is efficient for sparse graphs, is used to produce clusters that corresponds to social dimensions. The nodes of the original graph then are assigned weights along these social dimensions based on its incident edges. Then analysis and predictions can be done based on these social dimensions. The goal of our work is different in that we want to produce clusters that represent real communities, rather than a transformed feature space (though we can also use it for representing such social dimensions).

Correlational learning [15] aims to discover overlapping social groups in social networks supplemented with *tags* (or generally, labels specifying user interest) by first performing a singular value decomposition on the user-tag matrix, and the left and right singular vectors associated with the largest singular values (except the principle singular vector) are used as features in the latent space for users and tags, respectively. Then the similarity between two user-tag edges are defined

as a linear combination of the similarity between the two users and the two tags. Finally, the EdgeCluster k-means algorithm [14] is used to discover the social groups given the edge similarities. A drawback of this approach is that SVD computation is in general $O(mn^2 + n^3)$ where m and n are the number the users and tags, whereas BFG integrated with PIC is $O(|E|)$ where $|E|$ is the number of edges in the input graph. In addition, the proposed approach is able to cluster network data and data with arbitrary feature vectors.

"Path-folding" [13] is a technique similar to the matrix composition trick described in Section III-B, and is used to cluster large document collections. Here we use a different composition for clustering edges of a graph.

VI. Conclusion and Future Work

We proposed transforming a graph into a bipartite feature graph (BFG) as a general approach to apply graph partition methods such as spectral clustering to mixed membership clustering tasks. We show that a well-suited method is power iteration clustering (PIC), and when appropriately combined with BFG, it is able to show substantial improvement over single membership methods on even moderately mixed membership datasets.

An improvement to the proposed approach is learning the label assignment threshold parameter p based on some statistics or prior knowledge of category distribution. We can also extend this approach to apply BFG transformation to hypergraphs by clustering hyperedges. Lastly, we want to verify the generality of mixed membership clustering via edge clustering on a number of other efficient graph partition methods.

References

[1] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[3] A. Y. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14*, 2002.

[4] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[6] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *Journal of Machine Learning Research*, vol. 9, pp. 1981–2014, 2008.

[7] R. Balasubramanyan and W. W. Cohen, "Block-lda: Jointly modeling entity-annotated text and entity-entity links," in *The 11th SIAM International Conference on Data Mining*, 2011.

[8] F. Lin and W. W. Cohen, "Power iteration clustering," in *The 27th International Conference on Machine Learning*, 2010.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *The Annual Conference of the Special Interest Group on Data Communication*, 1999.

[10] J. Leskovec, "Dynamics of large networks," Ph.D. dissertation, Carnegie Mellon University, 2008.

[11] P. P. Talukdar, J. Reisinger, M. Paşca, D. Ravichandran, R. Bhagat, and F. Pereira, "Weakly-supervised acquisition of labeled class instances using graph random walks," in *The 2008 Conference on Empirical Methods in Natural Language Processing*, 2008.

[12] F. Lin and W. W. Cohen, "Adaptation of graph-based semi-supervised methods to large-scale text data," in *The 9th Workshop on Mining and Learning with Graphs*, 2011.

[13] ——, "A very fast method for clustering big text datasets," in *The 19th European Conference on Artificial Intelligence*, 2010.

[14] L. Tang and H. Liu, "Scalable learning of collective behavior based on sparse social dimensions," in *The 18th ACM Conference on Information and Knowledge Management*, 2009.

[15] X. Wang, L. Tang, H. Gao, and H. Liu, "Discovering overlapping groups in social media," in *The 2010 IEEE International Conference on Data Mining*, 2010.

[16] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford Digital Library Technologies Project, Tech. Rep., 1998.

[17] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[18] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, 2004.

[19] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[20] A. Condon and R. M. Karp, "Algorithms for graph partitioning on the planted partition model," *Random Structures and Algorithms*, vol. 18, no. 2, pp. 116–140, 2001.

[21] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, pp. 452–473, 1977.

[22] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, and E. S. andSteve M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003.

[23] A. Kale, A. Karandikar, P. Kolari, A. Java, T. Finin, and A. Joshi, "Modeling trust and influence in the blogosphere using link polarity," in *The International Conference on Weblogs and Social Media*, 2007.

[24] L. Adamic and N. Glance, "The political blogosphere and the 2004 u.s. election: Divided they blog," in *The WWW 2005 Workshop on the Weblogging Ecosystem*, 2005.

[25] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *The National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[26] Q. Lu and L. Getoor, "Link-based classification," in *The 20th International Conference on Machine Learning*, 2003.

[27] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, pp. 814–818, 2005.