CS 290H : Course Project

Dinesh Ramasamy
dineshr@ece.ucsb.edu

June 11, 2014

1 Introduction

In this project report, I will present the results of some computational experiments I performed in the context of my research on temporal evolution of topics in Twitter, in which I used spectral clustering techniques to summarize temporal activity. For the purpose of this course project, I specifically worked on improving the clustering algorithm by making the clustering algorithm hierarchical.

We first present the different candidate algorithms investigated, follow this up with results on synthetically generated graphs before concluding with some qualitative remarks on the performance on real-world data (specifically our Twitter data).

2 Notation

Consider the weighted undirected (connected) graph given by $G\left(V, E, w(\ ) :\ E \longmapsto \mathbb{R}^{+}\right)$. Let A denote the adjacency matrix for this weighted graph : $A(u,v) = A(v,u) = w(e)$ for $e = \{u,v\} \in E$ and $A(u,v) = A(v,u) = 0$ for $e = \{u,v\} \notin E$. Let $d(u)$ denote the degree of the vertex $u \in V$ given by $d(u) = \sum_v A(u,v)$. Let D^k denote the diagonal matrix with diagonal entries as $d^k(u)$. Let n denote the number of vertices $|V|$ and $m = \sum w(e)$, the total edge weight (m also satisfies $m = (1/2)\sum d(u)$).

3 Embeddings

The random walk on G is a Markov chain on the location of particle on the vertex set V given by the following state transition matrix:

$$\Pr\left[\text{Location}(t+1) = v | \text{Location}(t) = u\right] = \frac{w(u,v)}{d(u)},$$

where t is the time index. As we shall see, many embeddings of graphs derived from the spectrum of the normalized Laplacian matrix are motivated using the random walk on the graph.

3.1 Commute time embedding

The access time $h(u \to v)$ of the random walk of the vertex v from the vertex u is defined as the average number of hops taken by a random walk initiated at vertex u to reach vertex v. It is easy to see that access times must satisfy the following expression:

$$h(u \to v) = 1 + \sum_k \frac{w(u,k)}{d(u)} h(k \to v).$$

The commute time between a pair of vertices u and v is defined as the sum of access times $h(u \to v)$ and $h(v \to u)$. Using the above expression, it can be shown that commute time metric can be embedded in $\mathbb{R}^{n-1}$[2]:

$$\mathbf{m}(u) = \frac{1}{z_1(u)}\left[\frac{z_2(u)}{\sqrt{1-\lambda_2}}\ \frac{z_3(u)}{\sqrt{1-\lambda_3}}\ \cdots\ \frac{z_n(u)}{\sqrt{1-\lambda_n}}\right],$$

where $\mathbf{z}_k = [z_k(1); z_k(2); \cdots; z_k(n)]$ is the k-th largest eigen vector of the Normalized Adjacency Matrix $N = D^{-1/2}AD^{-1/2}$ (normalized so that $\mathbf{z}_k^T\mathbf{z}_k = 1$ and chosen to be orthonormal in the case of repeated eigen values) and λ_k the corresponding eigen values. The commute time between vertices u and v is given by $\|\mathbf{m}(u) - \mathbf{m}(v)\|^2$. Typically, the structure in the graph result in a decaying profile for the eigen values λ_k-s. As a result we may only use the first $L+1$-eigen pairs to arrive at an L-dimensional embedding of the graph in which distance squares approximately correspond to commute times between (corresponding) vertices.

3.2 Embedding of probability mass diffusion

Denoting the state transition matrix by P, we have that $P = AD^{-1}$, where $P(v,u)$ is the probability of a random walk at time $t+1$ moving to vertex v from vertex u at time t.

When we start the random walk at vertex u, the probability mass diffuses in the following manner: After k hops, the probability of finding the random walker in a particular vertex v is given by the v-th element of the vector $P^k\mathbf{e}(u)$, where $\mathbf{e}(u)$ is a vector all of whose entries are zero except the u-th element, which takes the value 1.

When we initiate S random walks at times $t = 0, \ldots, S-1$ from vertex u and observe the network at time S, the average number of particles we expect to see at each vertex network-wide is given by the corresponding element of $\mathbf{a}(u) = \sum_{k=1}^{k=S} P^k\mathbf{e}(u)$. This is an embedding of dimensionality at most $n-1$ (since the rank of N cannot exceed $n-1$). Since $P = D^{1/2}ND^{-1/2}$,

$$\begin{aligned}
\mathbf{a}(u) &= D^{1/2}\left(\sum_{k=1}^{k=S} N^k\right)D^{-1/2}\mathbf{e}(u) \\
&= D^{1/2}\left(\mathbb{I}_n - N\right)^{-1}\left(N - N^{S+1}\right)D^{-1/2}\mathbf{e}(u) \\
&= \frac{1}{z_1(u)\sqrt{2m}}\, D^{1/2}\left(\mathbb{I}_n - N\right)^{-1}\left(N - N^{S+1}\right)\mathbf{e}(u) \\
&= \frac{1}{z_1(u)\sqrt{2m}}\, D^{1/2}\sum_{k=2}^{k=n}\frac{\lambda_k - \lambda_k^{S+1}}{1-\lambda_k}\mathbf{z}_k\mathbf{z}_k^T\mathbf{e}(u) \\
&= \frac{1}{\sqrt{2m}}\sum_{k=2}^{k=n}\left(D^{1/2}\mathbf{z}_k\right) \times \left(\frac{\lambda_k - \lambda_k^{S+1}}{1-\lambda_k}\frac{z_k(u)}{z_1(u)}\right)
\end{aligned}$$

where we have used the fact that $z_1(u) = d(u)/(2m)$.

The motivation behind the above embedding is the following: The random walk is likely to stay with its "home cluster" for a reasonable amount of time after which $P^k\mathbf{e}(u)$ degenerates to the stationary distribution $\mathbf{d}/(2m) = \mathbf{z}_1^2$. Therefore we expect that members of the same cluster would see similar averaged probability diffusion vectors $\{\sum P^k\mathbf{e}(u)\}$.

As before the fast decay of λ_k-s allows for us to arrive at an L-dimensional approximation of this embedding using the first $L+1$ eigen pairs (also involves reducing the ambient dimensionality of the embedding using its basis $\{D^{1/2}\mathbf{z}_k : k = 2, \ldots, L+1\}$ - an operation of complexity $O(L^3 + Ln)$).

3.3 Laplacian embedding

Let $\{\mathbf{x}_k, \nu_k\}$ be the eigen pair of the Laplacian $L = \mathrm{diag}(\mathbf{d}) - A$ (indexed in increasing order of eigen values ν_k and $\mathbf{x}_k$ normalized so that $\mathbf{x}_k^T\mathbf{x}_k = 1$). The L-dimensional laplacian embedding of the vertex u is:

$$\mathbf{q}(u) = \left[\frac{x_2(u)}{\sqrt{\nu_2}}\ \frac{x_3(u)}{\sqrt{\nu_3}}\ \cdots\ \frac{x_{L+1}(u)}{\sqrt{\nu_{L+1}}}\right].$$

Motivation for normalization of random walk embeddings

We expect the *ordering* of commute times to capture cluster membership information. However, using the fact $\lambda_n \leq -1$ it can be shown that[2]:

$$0.5\left(\frac{2m}{d(u)}-1\right) \leq \|\mathbf{m}(u)\|^2 \leq \frac{1}{1-\lambda_2}\left(\frac{2m}{d(u)}-1\right), \text{ where } 2m = \sum_v d(v).$$

So the embedding of low degree nodes are pushed away from the origin while high degree nodes are pushed towards the origin. Therefore, even if their cluster membership is unambiguous, low degree nodes are far away from the high degree nodes inside the same cluster. Therefore, we expect that the direction $\mathbf{m}(u)/\|\mathbf{m}(u)\|$ of the embedding should have captured ordering of commute times. A similar argument can be made for the diffusion embeddings $\mathbf{a}(u)$ when the graph is regular. By normalizing the embeddings, we reveal this to the clustering algorithm which acts next on the embedding of vertices to cluster them.

4 Clustering algorithm

4.1 Deterministic Annealing (DA)

We employ Deterministic Annealing [3] to cluster the embeddings of the vertices $\mathbf{x}(u)$ ($\mathbf{x}(u)$ can be any of the above three embeddings or their normalized versions). Deterministic annealing tries to arrive at a best fit mixture of Gaussians model (GMM) for the collection of points $\{\mathbf{x}(u)\}$. The Gaussian mixtures are restricted to share the covariance matrix $(T/2)\mathbb{I}_L$. The unknown parameters are the means $\{\boldsymbol{\mu}(k)\}$ of the Gaussians. DA arrives at $\{\boldsymbol{\mu}(k)\}$ via EM iterates (alternating between refinement of cluster centers $\boldsymbol{\mu}(k)$ and cluster association probabilities $p(k|u)$). DA is a divisive algorithm that starts off with large values for the variance $T/2$. For large variances all cluster centers $\boldsymbol{\mu}(k)$-s coincide and we have effectively one cluster. As we lower the variance $T/2$, distinct clusters emerge (number of partitions increases). For a small enough variance $T/2$, we will have n distinct cluster centers $\{\boldsymbol{\mu}(u) = \mathbf{x}(u)\}$. At any intermediate stage of DA (any variance $T/2$), we can pause and ask for the maximum of $p(k|u)$ for each vertex u. This gives us a K-partition of the graph (where K is the number of distinct cluster centers $\{\boldsymbol{\mu}(k)\}$).

4.2 Partition goodness

We use the modularity score of the partition [1] to evaluate its quality. Let $\phi = \{S_1, \ldots, S_K\}$ be a K-partition of V. The modularity score of the subset $R \subseteq V$ is given by:

$$Q(R) = \left(\sum_{u,v\in R} \frac{A(u,v)}{2m}\right) - \left(\sum_{u\in R} \frac{d(u)}{2m}\right)^2.$$

The modularity score of the K-partition ϕ is given by $\sum_{l=1}^{l=K} Q(S_l)$. Note that $Q(V) = 0$ and $Q(S) > 0\ \forall S \subset V$.

4.3 Hierarchical DA for graph clustering

A straight forward application of DA for the graph clustering problem would be run DA until the number of clusters K exceeds a maximum value and pick the best partition ("goodness" quantified using modularity scores) encountered so far (when we do not know K - the correct number of clusters in the graph). Since all clusters share the same variance $(T/2)$, this results in poor performance for graphs with clusters of heterogenous sizes and intra-cluster connection densities.

Implemented for course project: To tackle this problem, we propose a hierarchical clustering algorithm: (i) Run DA - find the best partition. (ii) For each subset in a partition run DA and find the best partition (this can turn out be the whole subset itself). (iii) Repeat for all leaf nodes (subsets) until no new leaves emerge (that improve modularity).

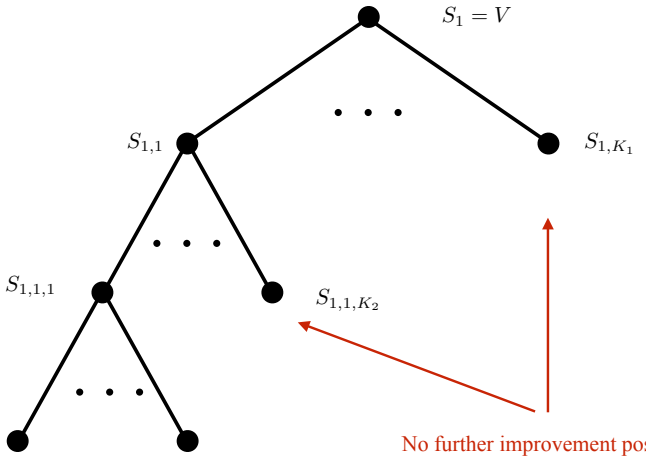

Figure 1: Hierarchical DA: Partition DA partitions recursively until no improvements can be made to $\sum Q(S_{\text{leaf}})$.

Similar ideas have been proposed in earlier graph clustering work (turns out that one of submission for DIMACS '10 graph clustering challenge uses similar ideas built using different graph bisecting primitive - not spectral DA).

5 Synthetic data

We generate graphs with clusters sizes $\{n_k\}$ and intra-cluster edge probability $\{p_k\}$. The inter-cluster edge probability is uniform and set to q.
Example one: We choose cluster sizes $n_k = [2000 + 200 \times (1:5),\ 20000 + 2000 \times (1:5)]$ and intra-cluster edge probabilities $p_k = 1.5\log(n_k)/n_k$. We set the inter-cluster edge probabilities to be $1.5\log(n)/n$, where $n = \sum n_k = 143000$ is the total number of vertices in the graph. We use a $L = 20$-dimensional embedding (with the spectrum computed using 200-iterations of Lanczos with reorthogonalization). Three example runs are plotted in Figure 2 in all of which we have identified the correct number of clusters. Further the labeling of vertices is also correct for a very high fraction of vertices (up to a permutation).
Example two: We change p_k to $1.1\log(n_k)/n_k$ and q to $1.75\log(n)/n$, while keeping the rest same as example one. These results are plotted in Figure 3.
Example three: We set p_k to $1.5\log(n_k)/n_k$ and q to $1.5\log(n)/n$ (same as example one), but reduce $n_k = [200 + 20 \times (1:5),\ 2000 + 200 \times (1:5)]$. These results are plotted in Figure 4. Note how the Laplacian embedding performs very poorly compared to the other two. Furthermore, for larger problems the Laplacian embedding seems to be less amenable to partition using DA.

6 UFL collection graphs - DIMACS '10 clustering challenge moderate sized graphs - vertices numbering a few hundred thousands

Results summarized in Table 1. Best known (a quick search) modularity scores for DIMACS10/coAuthorsDBLP and DIMACS10/coAuthorsCiteSeer (DIMACS '10 entries) hover around 0.85. Lots of parameters were chosen without optimization. So part of this gap could possibly be bridged.

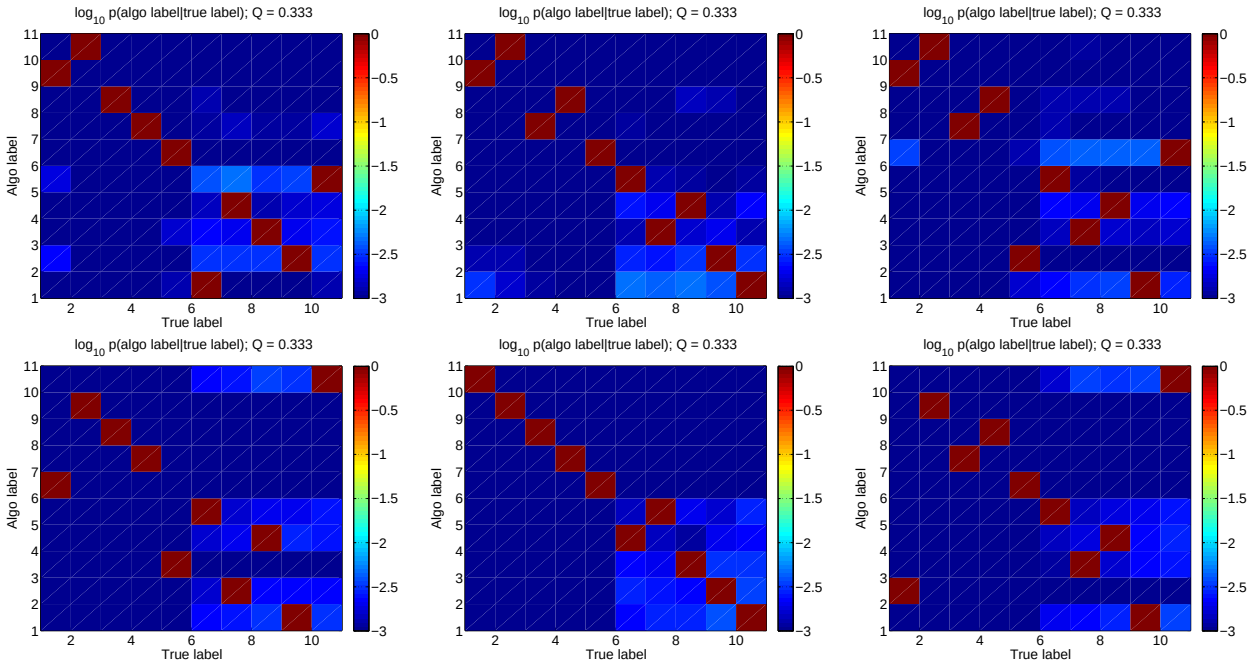

Figure 2: Six runs for example one. Top: Commute time embedding; Bottom: Diffusion embedding ($S = 10$ hops)

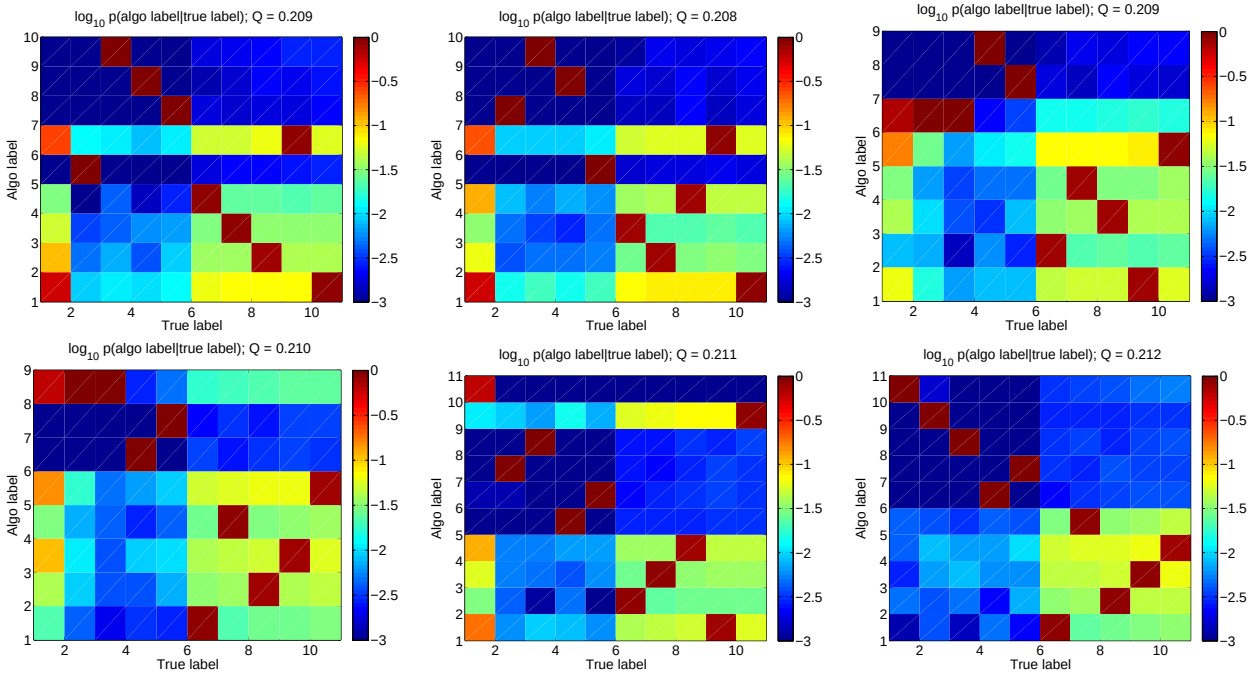

Figure 3: Six runs for example two. Top: Commute time embedding; Bottom: Diffusion embedding ($S = 10$ hops)

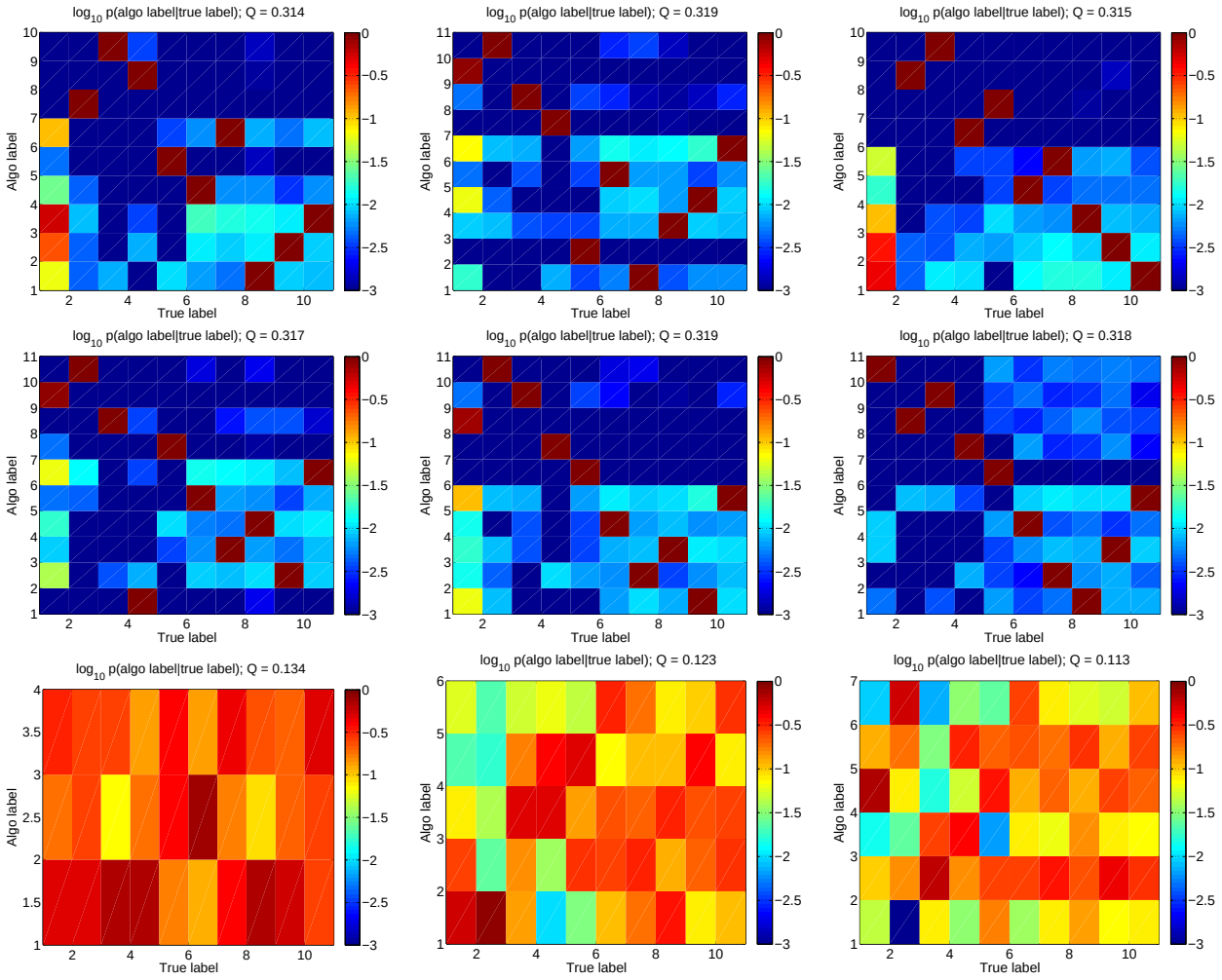

Figure 4: Three runs for example three (left to right are three graph instances). Top: Commute time embedding; Middle: Diffusion embedding ($S = 10$ hops); Bottom: Laplacian embedding (all normalized)

Name	n	m	Score (norm. comm)	Score (norm. diff 10-hops)
DIMACS10/coAuthorsDBLP	299067	1955352	0.5520	0.5516
DIMACS10/coAuthorsCiteSeer	227320	814134	0.6766	0.6493
DIMACS10/delaunay_n17	131072	786352	0.9054	0.8477

Table 1: UFL collection results – score is modularity. We use a $L = 30$ dimensional normalized commute-time embedding & normalized diffusion embedding ($S = 10$ hops) and we stop when the depth of the tree exceeds 20.

7 Conclusion

Embedding derived from the normalized laplacian ($\mathbb{I}_n - N$) seem to be more amenable for spectral clustering tasks.

References

[1] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5), Feb. 2010.

[2] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1):1–46, 1993.

[3] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11), Nov. 1998.