Sparse Matrix Operations On Social Networks

2020201026 | Mansi Khamkar | Group-9

Abstract This report provides information about the use of Linear Algebra in the field of graph theory and analysis. Precisely, it puts light on how sparse matrices represent the social network, and which important operations are performed on sparse matrices to explore and analyse social networks. These topics are explained in brief, among which I have covered Random Walks and Laplacian Matrix in more detail.

1. Introduction

Social networks are naturally modeled as graphs, which we sometimes refer to as a social graph. The entities are the nodes, and an edge connects two nodes if the nodes are related by the relationship that characterizes the network. There is a collection of entities that participate in the network. Typically, these entities are people, but they could be something else entirely. The best-known example of a social network is the "friends" relation found on sites like Facebook. Often, social graphs are undirected, as for the Facebook friends graph. But they can be directed graphs, as for example the graphs of followers on Twitter or Google+.

In the mathematical field of graph theory as well as for computations, a graph is modelled as a matrix. For a graph $G(V,E)$ with n vertices, the most popular form of matrix is the nxn adjacency matrix which represents the weight of the direct edge between any 2 vertices if they are adjacent, else 0 if not. A social graph is very huge with large number of vertices and each vertex (eg. a person) is directly linked to only a limited amount of vertices, which is very less than the total vertices in the graph. Hence, the adjacency matrix of the social graph is sparsely filled with non-zero entries. Such matrix, which has zero entries much more than the non-zero entries is called a sparse matrix.

2. Sparse Matrix

Sparse matrices are primarily populated with zeros. They are mainly used to represent loosely coupled systems where the the amount of significant data among entities is less. It is easy to visualize and analyse social graph represented by a sparse matrix. For example, the vertex (person) which has more number of non-zero entries in its row, means that he has more connections, and can be considered as an influencer. The sparsity of the matrix is given as:

$$\frac{\text{(total no. of elements)} - \text{(no. of non-zero elements)}}{\text{total no. of elements}}$$

2.1. Representation of Sparse Matrix

In the case of a sparse matrix, substantial memory requirement reductions can be realized by storing only the non-zero entries. Depending on the number and distribution of the non-zero entries, different data structures can be used and yield huge savings in memory when compared to the basic approach. The trade-off is that accessing the individual elements becomes more complex and additional structures are needed to be able to recover the original matrix unambiguously. Formats can be divided into two groups:

- Those that support efficient modification, such as DOK (Dictionary of keys), LIL (List of lists), or COO (Coordinate list).
- Those that support efficient access and matrix operations, such as CSR (Compressed Sparse Row) or CSC (Compressed Sparse Column).

Coordinate List: It stores a list of (row, column, value) tuples. Ideally, the entries are sorted first by row index and then by column index, to improve random access times. Only the non-zero elements must be in the list. The memory required here is: $3 * (no.\ of\ nonzero\ elements)$

Compressed Sparse Row / Yale format: It represents the matrix by three (one-dimensional) vectors, that respectively contain nonzero values, the extents of rows, and column indices by traversing the matrix row-by-row. The row vector is of size $n + 1$ that stores the cumulative number of non-zero elements upto $(i - 1)^{th}$ row. The memory required here is: $2 * (no.\ of\ nonzero\ elements) + m + 1$ where m is the number of rows.

Compressed Sparse Column: It represents the matrix by three (one-dimensional) vectors, that respectively contain nonzero values, the row index and the column pointer by traversing the matrix column by column. The column pointer vector stores the value indexes where each column starts. The memory required here is: $2 * (no.\ of\ nonzero\ elements) + n + 1$ where n is the number of rows.

2.2. Band Matrix

Band matrices are a form of sparse matrices whose non-zero entries are confined to a diagonal band, comprising the main diagonal and zero or more diagonals on either side. A matrix is called a band matrix or banded matrix if its bandwidth is reasonably small. Formally, consider an nxn martix B. All its elements are zero outside a diagonally bordered band whose range is determined by positive constants

$k_1 \& k_2$. Then the quantities k_1 and k_2 are called the lower bandwidth and upper bandwidth, respectively.

$$B_{i,j} = 0 \,.....\, if j < i - k_1 or j > i + k_2$$

Cuthill–McKee algorithm: The banded property corresponds to the fact that variables are not coupled over arbitrarily large distances. The Cuthill–McKee algorithm is used to permute a sparse matrix that has a symmetric sparsity pattern into a band matrix form with a small bandwidth. Using the graph adjacency matrix, the algorithm numbers the vertices according to a particular breadth-first traversal where neighboring vertices are visited and listed in order from lowest to highest vertex degree.

$$\begin{bmatrix} B_{11} & B_{12} & 0 & \cdots & \cdots & 0 \\ B_{21} & B_{22} & B_{23} & \ddots & \ddots & \vdots \\ 0 & B_{32} & B_{33} & B_{34} & \ddots & \vdots \\ \vdots & \ddots & B_{43} & B_{44} & B_{45} & 0 \\ \vdots & \ddots & \ddots & B_{54} & B_{55} & B_{56} \\ 0 & \cdots & \cdots & 0 & B_{65} & B_{66} \end{bmatrix}$$

FIGURE 1. *Example of a tridiagonal band matrix with bandwidth 1*

3. Sparse Matrix Multiplication

3.1. Matrix-Matrix Multiplication

Multiplication of two matrices $A\&B$ is computed as $C = AB^T$, which is defined only when the number of columnts of both $A\&B$ is same. We take the inner product of the i^{th} row of A, denoted as $A(i, :)$, and the j^{th} column of B^T, denoted as $B^T(:, j)$. Therefore, $C(i, j) = A(i, :).B^T(:, j)$. This approach computes one value at a time. To compute C all at once, we find the sum of outer products (rank-1 matrices) as:

$$C = \sum_r A(:, k).B^T(k, :)$$

Thus to compute $C = AB^T$, we first need to reconstruct matrix B^T into a dense matrix B^T_{dense} which is done by matrix coloring and then compute $Cdense = AB^T_{dense}$ such that the non-zero entries in C_{dense} are also non-zero entries needed to find C. Finally, C_{dense} is reorganized into C which is done by process similar to compression of B^T. Let $u, v \in R^n$. They are structurally orthogonal if $|u|^T|v| = 0$, then either $u(k) = 0 or v(k) = 0$or both. A set of vectors $u_1, ..., u_n$ is structurally orthogonal for $1 \le i,\ j \le n$. A set of q indices $l = l^1, ..., l^p$ for which $C(:, l_i)$ & $C(:, l_j)$ are structurally orthogonal when $1 \leq i,\ j \leq n \,\&\, i \neq j$. Matrix coloring $c = l_1, ..., l_p$ is a collection of index sets such that for $1 < k \leq n$, the index k appears in exactly one index set in $c$is an orthogonal set of C. We refer to an orthogonal set that is part of a coloring as a color. Applying coloring c to C produces matrix C_{dense} as follows:

$$C_{Dense}(j,k) = \begin{cases} C(j, l_k^r) & if\ C(j, l_k^r) \neq 0 \\ 0 & \sum_{i=1}^{qk} |C(j, l_k^i)| = 0 \end{cases}$$

3.2. Matrix-Vector Multiplication

Sparse Matrix-Vector multiplication (SpMV) is an important operation on sparse matrices. It is defined as: $\vec{y} = A\vec{x} + \vec{y}$ where A is a sparse matrix and $\vec{x}$ & $\vec{y}$ are column vectors. Whatever is the matrix dimensions, the number of floating operations is always more than thrice the number of non zero elements in A. The main aim of SpMV is to reduce the non uniform behaviour of matrix structure. This can be done by choosing a right matrix format and also implement the parallel kernel for each specific formats such it provides high efficiency. The well known formats are DIA, ELL, CSR and COO. The multiplication algorithm then loops through elements in these data structures to multiply and add them. For example, in CRS format the multiplication would be:

```
1: for $i = 0\ to\ m - 1$ do
2:     for $k = s_i\ to\ s_{i+1} - 1$ do
3:         add $x_k.A_{j,k}\ to\ y$
4:     end for
5: end for
```

4. Eigenvectors and Eigenvalues

Eigenvector or characteristic vector of a linear transformation is a nonzero vector that changes by a scalar factor when that linear transformation is applied to it. The corresponding eigenvalue, often denoted by λ is the factor by which the eigenvector is scaled. Geometrically, an eigenvector, corresponding to a real nonzero eigenvalue, points in a direction in which it is stretched by the transformation and the eigenvalue is the factor by which it is stretched. If the eigenvalue is negative, the direction is reversed.

$$Av = \lambda v \quad \Rightarrow (A - \lambda I)v = 0$$

Here, v is an eigenvector of the linear transformation of A and the scale factor λ is the eigenvalue corresponding to that eigenvector. I is the identity matrix and 0 is the zero vector. The above equation has a non-zero solution v if and only if the determinant of the matrix $(A - \lambda I)$ is zero. Therefore, the eigenvalues of A are values of λ that satisfy the equation $|A - \lambda I| = 0$.

4.1. Eigen Centrality Principle

Eigenvector centrality (also called eigen centrality or prestige score) is used in the analysis of social networks as a measure of the influence of a node in a network. All nodes in the network are assigned relative scores based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. A high eigenvector score means that a node is connected to many nodes who themselves have high scores. Along with how many people a person knows, it is also important to see whom he knows. The eigenvector centrality network metric takes into consideration not only how many connections a vertex has (i.e. its degree), but also the centrality of the vertices that it is connected to. Variants of eigenvector centrality are used to determine influence in social networks, for example, Google's PageRank algorithm. Also, one more example, social networks use the concept of Eigen Centrality to show the comment of the more influential people at the top.

At first, all nodes start off equal, but as the computation progresses, nodes with more edges start gaining importance. Their importance propagates out to the nodes to which they are connected. After re-computing many times, the values stabilize, resulting in the final values for eigenvector centrality.

Finding eigenvector centrality with adjacency matrix: For the given graph$G = (V, E)$ with $|V|$ denotes number of vertices. Let, A be the adjacency matrix. If vertex v is connected to vertex t, the relative centrality score can be defined as:

$$x_v = \frac{1}{\lambda} \sum_{t \in M(v)} x_t = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t$$

5. Random Walks on Social Network

Random Walks (RWs) are one of the most fundamental types of stochastic processes that lie at the core of many algorithms to uncover various types of structural properties of networks. RWs on networks are an example of a Markov chain in which the set of nodes is the state space and the transition probabilities depend on the existence and weights of the edges between nodes. They are often used to model numerous phenomena, including diffusion, interactions, opinions among humans and as a mechanism of search on social networks. For example: suggesting friends to a newcomer on social network. These processes would be optimal if one follows the shortest path between two nodes under considerations. However the shortest path can be found only after global connectivity is known at each node, which is improbable in practice. The random walk becomes important in the extreme opposite case where only local connectivity is known at each node.

In general, the underlying question in all these problems on social networks is: given a node in a graph, which other nodes are most similar to this node. Ideally we would like this proximity measure to capture the graph structure such as having many common neighbors or having several short paths between two nodes. A widely popular approach in graph-mining and machine learning literature is to compute proximity between nodes by using random walks on graphs: diffusion of information from one node to another.

5.1. Definition and Representation of Random Walks

A random walk on a network (graph) is a process that begins at some vertex, and at each time step moves to another vertex. When the graph is unweighted, the vertex the walk moves to is chosen uniformly at random among the neighbors of the present vertex. When the graph is weighted, it moves to a neighbor with probability proportional to the weight of the corresponding edge.

We consider an arbitrary finite undirected unweighted graph (network) $G = (V, E)$ which consists of nodes $i = 1, ..., n$ and links connecting them. We assume that the network is connected [i.e.,there is a path between each pair of nodes (i, j)], otherwise we simply consider each component separately. The connectivity is represented by the adjacency matrix A whose element Aij is nonnegative and denotes the weight on edge (i, j), and is zero if the edge does not exist. In unweighted graphs the weight of any edge is 1, whereas in a weighted graph it can be any positive number. Here, for undirected unweighted graph, the number of connected neighbors, called degree, is represented by a diagonal matrix D, where the diagonal elements Dii denote the degree of a node i (i.e. $d(i)$) and is calculated as $Dii = d(i) = \sum_j Aij$.

We define P to be the transition matrix of G (so Pij is the probability that we move from i to j in a step starting at i). These probabilities are calculated as:

$$P_{ij} = \begin{cases} 1/d(i) \ ..... \ if(i, j) \in E \\ 0 \ ..... \ otherwise \end{cases}$$

Using adjacency matrix:

$$P_{ij} = \frac{A_{ij}}{d(i)} = D^{-1}A_{ij}$$

The walker at node i and time t selects one of its connected neighbors with equal probability (as unweighted undirected graph) to which it hops at time $t + 1$. Let $\vec{p}$ be a probability distribution vector (value per node) where $p_i(t)$ is the probability that a walk is at node i at moment t. So, the random walk from node i at time t to node j at time $t + 1$ is:

$$p_j(t + 1) = \sum_i P_{ij}p_i(t) = \sum \frac{p_i(t)}{d(i)}A_{ij}$$

In matrix form: $$\vec{p}(t+1) = \vec{p}(t)P = \vec{p}(t)(D^{-1}A)$$

5.2. Stationary Distribution

Since G is connected, the random walk on G is a stationary stochastic process that corresponds to an irreducible Markov chain. When the walker keeps walking for a long time on a well behaved graph, there comes a point in time when the distribution does not change anymore. The stationary distribution at a node is related to the amount of time a random walker spends visiting that node. Well behaved graph is a graph which is irreducible (there is a path from every node to every other node) and aperiodic (the GCD (period) of all cycle lengths is 1). A stationary distribution of the random walk is a vector (probability distribution) $\vec{\sigma}$ which is unchanged by one step: i.e., $\vec{\sigma}$ such that $\vec{\sigma} = \vec{\sigma}P$. Equivalently, $\vec{\sigma}$ is an eigenvector for P with eigenvalue 1 (and since it's a probability distribution, $\sum_i \sigma_i = 1, \sigma_i \geq 0$).

Implications of the Perron Frobenius Theorem: If a markov chain is irreducible and aperiodic then the Perron-Frobenius theorem for nonnegative matrices implies that the largest eigenvalue of the transition matrix will be equal to 1 and all the other eigenvalues will be strictly less than 1. These results imply that for a well behaved graph there exists an unique stationary distribution.

Eventual convergence: Regardless of the initial probability distribution, a random walk on a well behaved graph always converges to the stationary distribution $\vec{\sigma}$. $\vec{\sigma}$ is defined as before to be the eigenvector of P with eigenvalue 1.
Proof: Let $\vec{v}$ be an arbitrary initial distribution. The eigenvectors of P form a basis $\omega_1, ...\omega_n$, so we can write $\vec{v}$ in terms of this basis: $\vec{v} = \sum_i \alpha_i \omega_i$. Hence,

$$\vec{v}P = \sum_i \alpha_i(\omega_i P) = \sum_i \alpha_i(\lambda_i \omega_i)$$

Iterating for t steps, $$\vec{v} \longrightarrow \vec{v}P^t = \sum_i \alpha_i(\lambda_i)^t \omega_i$$

For $i \neq 0$, we have $\lambda_i < 1$ and hence $(\lambda_i)^t \longrightarrow 0 \; as \; t \longrightarrow \infty$. These converge to 0 exponentially quickly, and the rate is determined by λ_2(the second largest eigenvalue). This rate of convergence is also known as mixing rate.

$$\lambda_1 = 1, \; so \; lim_{t\to\infty} \vec{v}P^t = \alpha_1\omega_1 = \sigma$$

Lemma: A well behaved graph has a stationary distribution proportional to the degrees of the vertices, since larger the degree of the node more likely a random walk is to come back to it.

$$\sigma(i) = \frac{d(i)}{2|E|}$$

6. Graph Laplacian Matrix

In the mathematical field of graph theory, the Laplacian matrix, also called the graph Laplacian, admittance matrix, Kirchhoff matrix or discrete Laplacian, is a matrix representation of a graph. The Laplacian matrix can be used to find many useful properties of a graph. It is used in various operations on huge graphs such as diffusion, partitioning, random walks, clustering, etc. Together with Kirchhoff's theorem, it can be used to calculate the number of spanning trees for a given graph. The sparsest cut of a graph can be approximated through the second smallest eigenvalue of its Laplacian by Cheeger's inequality. For example, We can get a good idea of the best way to partition a graph from the eigenvalues and eigenvectors of its Laplacian matrix. Given a simple graph G with n vertices, its Laplacian matrix L_{nxn} is defined as: $L = D - A$, where D is the diagonal degree matrix and A is the adjacency matrix of the graph. The elements of L are given by:

$$L_{i,j} = \begin{cases} deg(i) \text{ ..... } if\ i = j \\ -1 \text{ ..... } if\ i \neq j \text{ \& i is adjacent to j} \\ 0 \text{ ..... } otherwise \end{cases}$$

Symmetric normalized Laplacian
The symmetric normalized Laplacian matrix is defined as:

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

The elements for L^{sym} are given by:

$$L^{sym}_{i,j} = \begin{cases} 1 \text{ ..... } if\ i = j \text{ \& deg(i)} \neq 0 \\ -\frac{1}{\sqrt{deg(i).deg(j)}} \text{ ..... } if\ i \neq j \text{ \& i is adjacent to j} \\ 0 \text{ ..... } otherwise \end{cases}$$

Random walk normalized Laplacian
The random-walk normalized Laplacian matrix is defined as:

$$L^{rw} = D^{-1} L = I - D^{-1} A$$

The elements of L^{rw} are given by:

$$L^{rw}_{i,j} = \begin{cases} 1 \text{ ..... } if\ i = j\ \&\ deg(i) \neq 0 \\ -\frac{1}{eg(i)} \text{ ..... } if\ i \neq j \text{ \& i is adjacent to j} \\ 0 \text{ ..... } otherwise \end{cases}$$

The dimension of the nullspace of the Laplacian matrix equals the number of components of the corresponding graph. There exists a basis for the nullspace with each vector having components either zero or one. This provides a computational method to find the components of a graph: simply find a basis for the nullspace consisting of “binary" basis vectors. The magnitude of the off-diagonal entries of the Laplacian are either 0 or 1. We may wish to replace these integers with any number between 0 and 1 to indicate an affinity between two people. A technique called spectral clustering uses the Laplacian matrix to detect approximate clusters in these situations. The cluster detection problem is called cluster analysis and there are many approaches to the problem besides spectral clustering. If we zero-out the diagonal of the Laplacian matrix, we get (the negative of) a related object called the adjacency matrix of the graph. In the case of non-integer values, this is called the transition matrix. The spectral analysis of these matrices finds application in Markov chains, a multi-billion-dollar example of which is Google's PageRank algorithm.

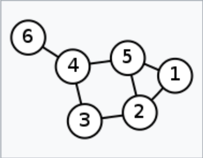
Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2&0&0&0&0&0\\ 0&3&0&0&0&0\\ 0&0&2&0&0&0\\ 0&0&0&3&0&0\\ 0&0&0&0&3&0\\ 0&0&0&0&0&1 \end{pmatrix}$	$\begin{pmatrix} 0&1&0&0&1&0\\ 1&0&1&0&1&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&1\\ 1&1&0&1&0&0\\ 0&0&0&1&0&0 \end{pmatrix}$	$\begin{pmatrix} 2&-1&0&0&-1&0\\ -1&3&-1&0&-1&0\\ 0&-1&2&-1&0&0\\ 0&0&-1&3&-1&-1\\ -1&-1&0&-1&3&0\\ 0&0&0&-1&0&1 \end{pmatrix}$

FIGURE 2. *Example of a labelled, undirected graph and its Laplacian matrix*

7. References

1. https://people.eecs.berkeley.edu/ aydin/GALLA-sparse.pdf
2. https://en.wikipedia.org/wiki/Sparse_matrix
3. https://www.mygreatlearning.com/blog/understanding-sparse-matrix/?amp
4. https://medium.com/datadriveninvestor/how-to-built-a-recommender-system-rs -616c988d64b2
5. https://en.wikipedia.org/wiki/Band_matrix
6. https://www.ibm.com/support/knowledgecenter/en/SSFHY8_6.2/reference/am5gr _bandma.html
7. https://en.wikipedia.org/wiki/Cuthill%E2%80%93McKee_algorithm
8. file:///home/mansi/Downloads/SIAM_Coloring2015.pdf
9. https://www.mcs.anl.gov/papers/P5007-0813_1.pdf
10. https://en.wikipedia.org/wiki/Matrix_multiplication
11. https://www.sciencedirect.com/science/article/pii/B9780128021187000273
12. https://cse.buffalo.edu/ erdem/cse331/support/matrix-vect/index.html
13. https://www.sciencedirect.com/topics/computer-science/eigenvector-centrality
14. http://www.sfu.ca/personal/archives/richards/Pages/wdr97.htm
15. https://sites.math.washington.edu/ morrow/336_11/papers/leo.pdf
16. https://www.cs.cornell.edu/courses/cs6850/2018sp/random-walks.pdf
17. https://cse.iitkgp.ac.in/ pawang/courses/SC15/rw1.pdf
18. http://infolab.stanford.edu/ ullman/mmds/ch10.pdf
19. http://infolab.stanford.edu/ ullman/mmds/ch10.pdf
20. https://web.math.ucsb.edu/ dls/Expository/GraphLaplacian.pdf
21. https://en.wikipedia.org/wiki/Laplacian_matrix

8. Team Contribution

- Vaishali Singh(2020201070)
  - Introduction & representations to Sparse matrix

- B. Sindhu(2020201048) & Mahak Modani(2020201068)
  - Matrix Multiplication

- Abhishek Sharma(2020201055)
  - Band Matrix

- Mansi Khamkar (2020201026)
  - Random Walks on social network and Laplacian Matrix

- Tushar Bhatt (2020201081)
  - Physical Significance of sparse matrices

- Shubham Swetank(2020201025)
  - Eigen Centrality