YOUR HOMEWORK ASSIGNMENT
Editor: Nargess Memarsadeghi, nargess.memarsadeghi@nasa.gov

Social Resistance

Michael P. Friedlander | University of British Columbia
Nathan Krislock | Northern Illinois University
Ting Kei Pong | Hong Kong Polytechnic University

This case study concerns an important question in social networks: How closely related are two people in a network? We measure the strength of the relationship between two people in terms of their *resistance distance*, a notion of distance that has its origin in the study of electrical networks.[1] In that context, the resistance distance measures the electrical resistance between the two nodes in a network, assuming unit resistance on each edge. Its relationship with the information distance in social networks was recently identified.[2,3]

For large networks, this distance measure can be expensive to compute. As the network changes, it's useful to have an inexpensive way to make updates. We'll study how various techniques for updating matrix factorizations can help efficiently compute the resistance distances with small changes in the network. We focus on Cholesky rank-one updates, and also briefly discuss the Sherman-Morrison-Woodbury formula (as one of the last exercises). We expect that you're familiar with the idea of the Cholesky factorization of positive definite matrices.[4] We'll guide you through a derivation of the formula for the Cholesky rank-one update, and introduce two aspects that may be new for you. First, you'll have to figure out the vector used for the rank-one update when one additional connection in the network is identified. Second, when matrices are large, you'll observe that there can be a big difference in computational effort when the computation involves matrix-matrix multiplications rather than only matrix-vector multiplications.

Zip File

Download this companion zip file, which contains the code and data you'll use while working through the case study, to the directory `social-resistance`: unzip('http://www.cs.ubc.ca/~mpf/social-resistance.zip','social-resistance');

Make sure that the directory `social-resistance` is in the Matlab path, or execute the command `addpath('social-resistance')`.

As you read through this case study, you'll be asked to work on a series of activities, including short mathematical and programming exercises, to try out these ideas on real data. You'll need a working copy of Matlab to complete the programming activities. Figure 1 is a good place to start visualizing the core issues.

Background

In a social network, the question of how "close" two people (or groups) are is of great importance. The notion of closeness makes this question tricky. Take the network structure induced by Facebook friendship as an example. Such a structure can be conveniently modeled using an undirected graph. Mathematically, an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ that describes connections between pairs of nodes. To model friendship on Facebook, we can represent an individual by a node and join two nodes with an edge if these two people are friends on Facebook. Figure 2 shows a small undirected graph representing the friendship of six individuals.

A natural measure of closeness in a graph is the length of the shortest path between two nodes. This is the least number of edges that need to be traversed to get from one node to another. Though well-studied mathematically, this notion isn't an appropriate one when it comes to measuring how close two people are on a social network. For instance, according to this measure, Jane is equally close to Alfred and Felix in Figure 2. Intuitively, however, we would expect Jane to be closer to Felix because they have two common friends. Similarly, according to the shortest path measurement, as immediate friends, Dave is equally close to Jane and Felix. But again, we should expect Dave to be actually closer to Jane because they share two common friends. Intuition tells us that our closeness measurement should obey the following rule:

> Two individuals are closer if they have more common friends.

 1521-9615/16/$33.00 © 2016 IEEE Copublished by the IEEE CS and the AIP

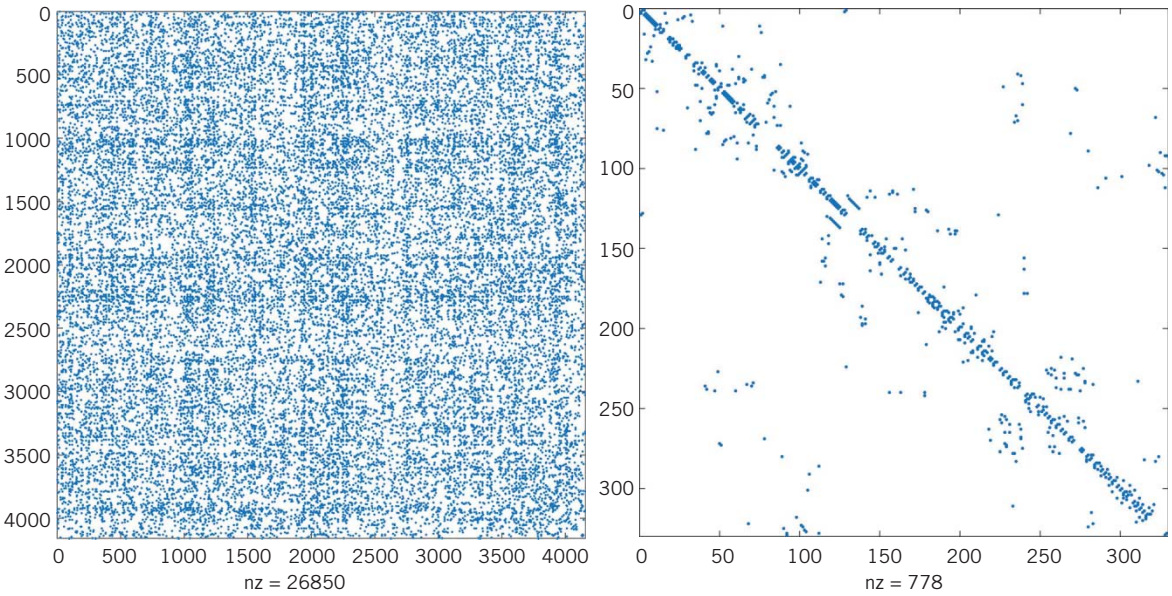

Figure 1. An author collaboration network (left) and a traffic network (right). The patterns in these figures are strikingly different: there seems to be relatively little structure in the collaboration network, as compared to the highly structured traffic network.

The resistance distance conforms with this intuitive definition.

Resistance Distance

To describe the resistance distance, we need more notions from graph theory. We can conveniently represent an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes using an $n \times n$ symmetric matrix A, which is called the *adjacency matrix* of the graph. The entries of this matrix are defined by

$$A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases},$$

where we use (i, j) to denote an edge between nodes i and j, and the set $\mathcal{E}$ contains a list of all edges in the graph. For example, the adjacency matrix of the graph in Figure 2 is given by (the letters along the top are abbreviations for the full names)

$$A = \begin{array}{c} \begin{matrix} A & J & L & D & E & F \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} \text{Alfred} \\ \text{Jane} \\ \text{Lucy} \\ \text{Dave} \\ \text{Erica} \\ \text{Felix} \end{matrix}$$

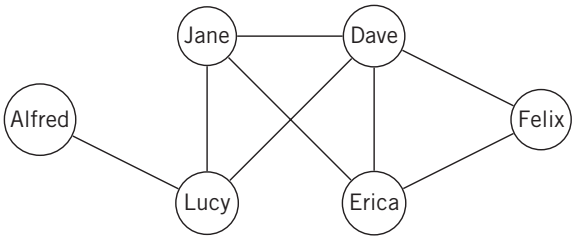

Figure 2. A small social network of six individuals.

and the sets of nodes and edges are given by

$$\mathcal{V} = \{\text{Alfred, Jane, Lucy, Dave, Erica, Felix}\},$$

$$\mathcal{E} = \{(\text{Alfred, Lucy}), (\text{Jane, Lucy}), (\text{Jane, Dave}), (\text{Jane, Erica}), (\text{Lucy, Dave}), (\text{Dave, Erica}), (\text{Dave, Felix}), (\text{Erica, Felix})\}.$$

Note that this graph is *undirected*, that is, it doesn't distinguish between the edges (i, j) and (j, i).

Let d be the vector whose ith entry is the degree of node i, that is, the number of edges attached to node i. Then $d = Ae$, where e is the vector of all ones. In our example, $d = (1, 3, 3, 4, 3, 2)$. The Laplacian matrix L is defined as

$$L := \text{Diag}(d) - A,$$

YOUR HOMEWORK ASSIGNMENT

where Diag(d) is the diagonal matrix having the vector d along its diagonal. For the graph in Figure 2,

$$L = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 3 & -1 & -1 & -1 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 \\ 0 & -1 & -1 & 4 & -1 & -1 \\ 0 & -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}.$$

The Laplacian matrix is a symmetric matrix and is always positive semidefinite—that is, $x^T L x \geq 0$ for all vectors x.

A graph is called *connected* if any two nodes in the graph are connected by a set of edges. For example, the graph in Figure 2 is connected; however, if the edges (Jane, Lucy) and (Dave, Lucy) are removed, then the resulting graph isn't connected because there would be no path from Alfred to Felix, for example. The nullspace for the Laplacian matrix of a connected graph is the set of vectors that are constant, that is, $Lv = 0$ if and only if $v = \lambda e$ for some scalar λ. You will explore these properties in Activity 1, below.

ACTIVITY 1

1. Prove that

$$L = \sum_{(i,j)\in E} (e^{(i)} - e^{(j)})(e^{(i)} - e^{(j)})^T, \quad (1)$$

   where $e^{(i)}$ is the vector that is one in the ith entry, and is zero in every other entry.
2. Use the relationship in Equation 1 to show that the Laplacian matrix is positive semidefinite.
3. Show that $Le = 0$.
4. Let G be a connected graph and let L be its Laplacian matrix. Show that $L + ee^T$ is positive definite, that is, that $x^T(L + ee^T)x > 0$ for all nonzero vectors x.

The resistance distance between nodes i and j on a connected graph is defined as

$$r_{ij} := \|e^{(i)} - e^{(j)}\|_M^2,$$

where $M = (L + ee^T)^{-1}$, and $\|x\|_M$ is the weighted 2-norm of a vector x, defined by

$$\|x\|_M = \sqrt{x^T M x}.$$

As required for $\|\cdot\|_M$ to be a valid vector norm, M is positive definite because $L + ee^T$ is positive definite. Note that the usual 2-norm of a vector corresponds to $M = I$, that is, $\|x\| \equiv \|x\|_I = \sqrt{x^T x}$.

ACTIVITY 2

For the network in Figure 2, compute the resistance distances between Alfred and Jane, Alfred and Felix, Dave and Jane, as well as Dave and Felix. Does this notion of distance conform with our intuition?

Our next activity computes the resistance distances in a larger network: an author collaboration network on arXiv on the subject of "General Relativity." The data comes in the form of an adjacency matrix: the (i, j)th entry is 1 if person i and person j are coauthors, and zero otherwise.

ACTIVITY 3

1. Included in the zipped file is the matrix file Graph.mat (a modified version of the matrix SNAP/ca-GrQc from the UF Sparse Matrix Database[5]). This matrix is an adjacency matrix of the collaboration network in Figure 1. Use the following Matlab commands to load and view the matrix, as in Figure 1:

```
load Graph
spy(A);
```

2. Let $L + ee^T = U^T U$ be the Cholesky factorization of $L + ee^T$, where U is upper triangular. Show that the resistance distances can be computed as

$$r_{ij} = \|U^{-T}(e^{(i)} - e^{(j)})\|^2,$$

   where $U^{-T} = (U^{-1})^T = (U^T)^{-1}$. Based on this, write a Matlab function to compute r_{ij} for any given i and j. To compute $z = U^{-T}(e^{(i)} - e^{(j)})$ efficiently, you should solve the lower-triangular system $U^T z = b$, with $b = e^{(i)} - e^{(j)}$, using forward substitution; this is done in Matlab using the command `z = U'\b`. Don't be tempted to use the Matlab command `z = inv(U')*b`! This is less efficient and less accurate.

```
[rij,U] = Resistance(Lones,i,j);
```

   The inputs are the positive definite matrix `Lones` (corresponding to $L + ee^T$ in the problem), i and j. The first output argument `rij` is the resistance distance r_{ij}, and the second output argument `U` is the Cholesky factor of `Lones`. You can use the Matlab built-in function `chol` to compute U. Compare your output against the solution file `Resistance_soln.p`.
3. Complete the missing sections of code in the file `changeResistance.m`:

```
r = changeResistance(A,i,j,k);
```

   This code takes in the adjacency matrix A, i, and j, and also a positive integer k. It connects i to at most k random neighbors of j one by one, and recomputes the

resulting resistance distance between i and j after each addition. The k computed resistances are recorded in the output vector r.

Experiment with the values $i = 500$, $j = 3000$, and $k = 10$. Compare your output against the solution `changeResistance_soln.p`. You can use `rng('default')` to reset the random seed for comparing the two codes.

Plot the decrease in the resistance distance as k increases. This indicates, intuitively, if researcher i works with coauthors of researcher j, then i and j get "closer" to each other on this collaboration network.

Rank-One Updates

In the above activity, we performed a Cholesky factorization each time a collaboration is added. The overall work can be significantly reduced by using a Cholesky rank-one update, as we describe next.

Suppose that B is an $n \times n$ positive definite matrix and $B = U^T U$ is its Cholesky factorization. Given this upper triangular matrix U and a vector x, we would like to obtain a Cholesky factorization for $B + xx^T$. Note that

$$B + xx^T = U^T U + xx^T = \begin{bmatrix} U^T & x \end{bmatrix} \begin{bmatrix} U \\ x^T \end{bmatrix},$$

where the matrix

$$\begin{bmatrix} U \\ x^T \end{bmatrix} \tag{2}$$

is of size $(n + 1) \times n$.

Activity 4

Let a and b be given scalars, with $a > 0$. Find (c, s) with $c > 0$ and $c^2 + s^2 = 1$ so that

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sqrt{a^2 + b^2} \\ 0 \end{pmatrix}. \tag{3}$$

The 2-by-2 matrix in Equation 3 is an example of a Givens rotation. One way to perform a Cholesky rank-one update is to apply successive Givens rotation matrices Q to "zero out" the last row of the matrix in Equation 2 step by step and obtain a new U in the process: the Cholesky factor of $B + xx^T$.

Using the formulas you discover in Activity 4, you can find (c, s) with $c > 0$ and $c^2 + s^2 = 1$ such that

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{pmatrix} U_{11} \\ x_1 \end{pmatrix} = \begin{pmatrix} \sqrt{U_{11}^2 + x_1^2} \\ 0 \end{pmatrix}.$$

Then let Q_1 be the Givens matrix

$$Q_1 = \begin{bmatrix} c & 0 & -s \\ 0 & I_{n-1} & 0 \\ s & 0 & c \end{bmatrix},$$

where I_{n-1} is the $(n - 1) \times (n - 1)$ identity matrix, and 0 is used to represent a vector or matrix of zeros with the appropriate dimension. Multiply the matrix in Equation 2 by Q_1, and "zero out" the first entry of the last row:

$$Q_1 \begin{bmatrix} U \\ x^T \end{bmatrix} = Q_1 \begin{bmatrix} U_{11} & * \\ 0 & * \\ x_1 & * \end{bmatrix} = \begin{bmatrix} \sqrt{U_{11}^2 + x_1^2} & * \\ 0 & * \\ 0 & * \end{bmatrix} =: \begin{bmatrix} U^{(1)} \\ (x^{(1)})^T \end{bmatrix}.$$

The symbol * is used as a wild card to denote a possibly nonzero entry. Because of the position of I_{n-1} in Q_1, it only affects the first and last rows of the matrix that it multiplies.

Activity 5

1. Verify that Q_1 is orthogonal, that is, show that $Q_1^T Q_1$ is the identity matrix of size $(n + 1)$. Then show that

$$\begin{bmatrix} U^{(1)} \\ (x^{(1)})^T \end{bmatrix}^T \begin{bmatrix} U^{(1)} \\ (x^{(1)})^T \end{bmatrix} = B + xx^T.$$

A matrix Q_2 is then similarly chosen to operate on the second and last rows of

$$\begin{bmatrix} U^{(1)} \\ (x^{(1)})^T \end{bmatrix}$$

so as to zero out the second entry of the last row, and so on. Thus, after k iterations we have

$$\begin{bmatrix} U^{(k)} \\ (x^{(k)})^T \end{bmatrix} = Q_k \cdots Q_2 Q_1 \begin{bmatrix} U \\ x^T \end{bmatrix}.$$

Complete the missing sections of code in the file `mycholupdate.m`. This function takes the above approach to compute the Cholesky factor of $B + xx^T$, given U and x.

```
U = mycholupdate(U,x);
```

Remember, you don't need to form the whole Givens matrix; in each iteration, you only need to find (c, s) with $c > 0$ and $c^2 + s^2 = 1$ such that

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{pmatrix} U_{ii}^{(i-1)} \\ x_i^{(i-1)} \end{pmatrix} = \begin{pmatrix} \sqrt{(U_{ii}^{(i-1)})^2 + (x_i^{(i-1)})^2} \\ 0 \end{pmatrix}.$$

Test your code with random instances:

```
B = randn(3000); B = B'*B;
U = chol(B); x = randn(3000,1);
```

Compare your code against the solution `mycholupdate_soln.p`, and then compare your solution to Matlab's built-in function `cholupdate.m`.

2. Complete the missing sections of code in the file `changeResistance_r1.m`:

```
r = changeResistance_r1(A,i,j,k);
```

It has the same inputs and outputs as `changeResistance.m`, and it uses the Cholesky rank-one update in the loop instead of computing Cholesky factors afresh. Guided by Activity 1.1, you'll need to figure out what the x is when an edge is added to the graph.

Experiment with your code using $i = 500$, $j = 3000$, and $k = 10$. Compare your output against the solution file `changeResistance_r1_soln.p`. How does it compare with the method in Activity 3.3 in terms of speed? You can also try to use the Matlab built-in function `cholupdate.m` in place of `mycholupdate.m`.

The Sherman-Morrison-Woodbury Formula

Another widely used method for obtaining $(B + xx^T)^{-1}$ given B^{-1} is the Sherman-Morrison-Woodbury (SMW) formula. The SMW formula states that

$$\left(B + xx^T\right)^{-1} = B^{-1} - \frac{1}{1 + x^T B^{-1} x} B^{-1} x x^T B^{-1}.$$

The formula is particularly useful in our scenario because we only need to find the matrix-vector product $(B + xx^T)^{-1}v$ for a suitable B, x, and v.

Activity 6

1. Argue that if B^{-1} is given, then computing $(B + xx^T)^{-1}v$ for given vectors x and v using the SMW formula requires at most Cn^2 additions and multiplications, for some constant C independent of n.

2. Complete the missing sections of code in the file `chol_vs_smw.m`. This function computes

$$y = \left(L + ee^T + xx^T\right)^{-1}\left(e^{(i)} - e^{(j)}\right),$$

where x corresponds to adding an edge between node i and a random neighbor of j:

```
[errchol,errsmw] = chol_vs_smw(A,i,j);
```

In this code, we assume knowledge of the Cholesky factor U of $L + ee^T$, and compare the additional cost needed to obtain y from performing the Cholesky update or using the SMW formula. The outputs `errchol` and `errsmw` are residuals

$$\left\|\left(L + ee^T + xx^T\right) y - \left(e^{(i)} - e^{(j)}\right)\right\|,$$

with the y obtained from the corresponding approach.

Compare your code against the solution file `chol_vs_smw_soln.p`. You can also use Matlab's built-in function `cholupdate.m` in place of `mycholupdate.m`.

The SMW formula is convenient, but for some matrices, it can be unstable—that is, roundoff errors intrinsic to a computer's finite-precision arithmetic can overwhelm the final calculated answer.[6] It doesn't happen often, but it's something to bear in mind.

In our last activity, we're going to apply what we learned above to analyze the traffic network shown in the panel on the right-hand side of Figure 1.

Activity 7

1. Inside the zip file is the matrix file Graph2.mat, which is a modified version of the matrix SNAP/roadNet-CA from the UF Sparse Matrix Database.[5] This is the adjacency matrix of the traffic network in Figure 1 and represents a subset of the road network of California. Load and view the matrix as in Activity 3.1.

   Compute the resistance distances for all pairs of nodes. What's the average resistance distance across the whole network? Plot a histogram of the resistance distances.

2. As a greedy approach to decrease the average resistance distance, you can build a new road between the pair of nodes i and j that have the largest resistance distance.

   What's the average resistance distance across the whole network after such a new road is built? You can use either the Cholesky rank-one update or the SMW formula to compute the new resistance distances. You might also want to test the effectiveness of the greedy approach by comparing the resulting average resistance distance with that obtained by adding a new road between two randomly chosen nodes.

We've only touched on a few of the basic methods in numerical linear algebra needed to analyze networks. With just a few refinements, it's possible to apply the techniques that you've learned to much larger networks. Can you spot opportunities to make your code more efficient?

For example, in Activities 3 and 6, you computed the Cholesky factorization of $L + ee^T$ by first explicitly forming this matrix. But the very same techniques you studied for updating the Cholesky factorization could be used: first, compute the Cholesky factorization $U^TU = L$, and then obtain the required factorization of $L + ee^T$ via `U = mycholupdate(U,e)`.

We also haven't touched on a crucial property of most networks that arise in practice: they have relatively few edges as compared to nodes, which means that the adjacency and Laplacian matrices that we studied are sparse (that is, they have very few nonzero entries). The traffic network from which Figure 1 is extracted exemplifies this property. The original network, which has 19,171,281 nodes, is too large to treat with the dense-matrix techniques used in this case study but could be easily stored and manipulated by taking advantage of sparsity.

Numerical linear algebra is a cornerstone of countless practical problems, both old and new. We hope this case study has piqued your interest!

Acknowledgments

We extend sincere thanks to Dianne O'Leary and Nargess Memarsadeghi for their steady encouragement and their many thoughtful suggestions for improving the presentation. This work was funded in part by a grant from the Carl Wieman Science Education Initiative at the University of British Columbia.

References

1. D. Klein and M. Randic, "Resistance Distance," *J. Mathematical Chemistry*, vol. 12, 1993, pp. 81–95.
2. E. Bozzo and M. Franceschet, "Resistance Distance, Closeness, and Betweenness," *Social Networks*, vol. 35, no. 3, 2013, pp. 460–469.
3. K. Stephenson and M. Zelen, "Rethinking Centrality: Methods and Examples," *Social Networks*, vol. 11, no. 1, 1989, pp. 1–37.
4. L.N. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, 1997.
5. T.A. Davis and Y. Hu, "The University of Florida Sparse Matrix Collection," *ACM Trans. Mathematical Software*, vol. 38, no. 1, 2011, pp. 1:1–1:25; www.cise.ufl.edu/research/sparse/matrices.
6. E.L. Yip, "A Note on the Stability of Solving a Rank-p Modification of a Linear System by the Sherman-Morrison-Woodbury Formula," *SIAM J. Scientific and Statistical Computing*, vol. 7, no. 2, 1986, pp. 507–513.

Michael P. Friedlander is a professor of computer science and mathematics at the University of British Columbia. His research is primarily in developing and implementing numerical methods for large-scale optimization. Friedlander serves on the editorial boards of SIAM J. Optimization, SIAM J. Matrix Analysis and Applications, Mathematics of Operations Research, and Mathematical Programming. He received a PhD in operations research from Stanford University. Contact him at mpf@cs.ubc.ca.

Nathan Krislock is an assistant professor at Northern Illinois University. His research interests include continuous and combinatorial optimization, with a focus on semidefinite programming, numerical computation, and applications. Krislock received a PhD in mathematics from the University of Waterloo. Contact him at nkrislock@niu.edu.

Ting Kei Pong is an assistant professor at Hong Kong Polytechnic University. His research is mainly on continuous optimization, with a focus on first-order methods for large-scale problems. Pong received a PhD in mathematics from the University of Washington. Contact him at tk.pong@polyu.edu.hk.