Computational Proposal for Multi-Circle Mixed Model

avrahami.ben

April 2020

1 Introduction

In this document, we will examine and propose a model for the requirement of computing a multi-circle mixed-model infection step.

2 Mathematical Requirement

Given a population size N, the client requires the following structures and operations:

- Set of encounter kinds K, each encounter kind will possess the following attributes
  - Real matrix M, of size $N \times N$, denoting the probability of two agents meeting in a step
    - Sparse: we expect the grand majority of cells in M to have value 0. So much so that we can assume a number $T \ll N$ exists, and that there exists no row or column with more than T non-zero values.
    - Symmetric, it is as of yet unknown whether M is necessarily symmetric.
  - Real matrix W, of size $N \times N$, denoting an arbitrary value relating to infection given a meeting.
    - M-sparse: for any coordinate $i, j \in N$, $W[i, j] = 0 \leftrightarrow M[i, j] = 0$, barring any scalar row/column modifications.
    - Symmetric, it is as of yet unknown whether M is necessarily symmetric.
    - multiply row/column by scalar: given a row/column i and a scalar factor $f \in [0, 1]$, we want all the values in the row/column to multiply by the factor. In case that $f = 0$, the operation should be reversible.

    * add non non-zero elements row/column by constant: given a row/column i and a constant $f \in \mathbb{R}$, we want all the values in the row/column to add by the constant, where the value is non-zero.
  - Function $F : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ s.t.

$$w = 0 \vee v = 0 \rightarrow F(w, v) = 0$$

    at this time, it is useful to define

$$F^* = 1 - F$$

  - Operation: Randomly generate real $N \times N$ matrix R s.t.

$$\forall_{i,j \in [0,N]} \qquad \begin{matrix} P[R_{i,j} = W_{i,j}] = M_{i,j} \\ P[R_{i,j} = 0] = 1 - M_{i,j} \end{matrix}$$

    * POA: given R and real vector v of size $1 \times N$, we would like to compute, for each row r:

$$1 - \prod_{j \in (0..N-1)} (1 - F(R[r, j], v[j]))$$

      we can also assume that v is sparse, s.t. there exists $L \ll N$ non-zero values in v. For all rows in R, this results in real vector U.
      in truth, we will calculate $\prod_{j \in (0..N-1)} F^*(R[r, j], v[j])$, which enters into a vector U^*, which is equal to $1 - U$
    * Set-Aside stats: for each j, $\{i | R^k_{i,j} \neq 0\}$ and $\sum U^k = (N - \sum U^{*k})$ must be set-aside for future use (at least for a known number of steps).
  - operation : Given a new M' matrix (possibly of different implementation than M), the kind's M matrix should swap with M'. W should/can change accordingly to maintain M-sparcity.

- MPOA: given a set of vectors $\{U\}_0^K$ and V vector, we would like to calculate an infection chance vector C s.t.

$$C_i = \left\{ \begin{array}{cc} \text{undefined} & \text{if } V_i \neq 0 \\ 1 - \prod_{i \in (0..K-1)} (1 - U_i^k) & \text{otherwise} \end{array} \right\}$$

  In truth, if given the set $\{U^*\}_0^K$, we would instead calculate

$$C_i = \left\{ \begin{array}{cc} \text{undefined} & \text{if } V_i \neq 0 \\ 1 - \prod_{i \in (0..K-1)} U^{*k}_i & \text{otherwise} \end{array} \right\}$$

3 Implementation

Dense matrix implementation We will use a slightly modified version of the parasymbolic matrix as presented in previous documents. The POA implementation will remain as before. Generation of R matrix is trivial.

Immutability After initial generation, we don't expect M or W matrices to change, save for the add/multiply row/column by scalar and M-Swap operations.

4 Dense Implementation

Given that some kinds will have M and W matrices that is clustered, we can divide it into a set of dense matrices. Formally:

$$\begin{gathered} Q_0^h | Q_0 \oplus Q_1 \cdots \oplus Q_{N-1} = [0..N-1] \\ \text{s.t.} \\ \forall_{i,j \in [0..N-1]} M_{i,j}^k \neq 0 \rightarrow \exists Q_g | i, j \in Q_g \end{gathered}$$

In these cases, the matrices M^k, W^k, and R^k can appear as sparse matrices, but will be implemented as a list of dense matrices, each one of shape $Q_i \times Q_i$. Alongside a $1 \times N$ array detailing to which sub-matrix each value is stored, similar capabilities can be achieved, alongside the major improvements of using numpy matrices.