Mizuhiro Suzuki

June 20, 2020

Contents

1

- Main Matlab file: `"GMMDiffusion/Main_models_1_3"` for model without λ (endorsement effects) and `"GMMDiffusion/Main_models_2_4"` for model with λ

2 `Main_models_1_3.m`

Part 0

- 26: 2 model types: modelType == 1 for $q_N = q_P$ and == 3 for $q_N \neq q_P$

Part 2

- 33: Different versions of moments (In the paper, they say they use 6 moments, but no version says 6 moments, why?)

Part 3

- 49: `TMonths`: number of months in data of each village

Part 4

- 63-69: Parameter grids: although explained in detail in the Supplementary Materials, not shown how they did it

Part 5

- 90: Load the network data, which contains adjacency matrices of 43 villages
- 95-: For each village,
    - 101: Load the leader data: ID and dummy of being a leader or not
    - 106: Load the take-up data: Only dummy (maybe the same order as the one above?)
    - 107: `EmpRate` is the proportion of non-leaders who took up MF
    - 110: `inGiant`??? Maybe the biggest cluster in the village?
    - 113-114: `d` = degree of network, `hermits` = households without any connections with other households (?)
    - 117: Load the covariates of the households (not sure which column corresponds to which variable, maybe in readme file or somewhere?)
    - 120: Select covariates to use (but in the end they used all the covariates in the data?)
    - 123-130: Restrict the sample to households belonging to the giant networks?
        - 123: Leaders
        - 124: Take-up
        - 125: Covariates
        - 127: Take-up by leaders
        - 128: Covariates of leaders
        - 129: Outcome = Take-up by leaders
        - 130: Covars = Covariates of leaders
            - `Outcome` and `Covars` will be used to estimate β
    - 133-137: Second neighbors (`sec`) = matrix indicating households whose distance between them is 2. Here, X^2 means $X * X$. (See here)
        - 134-136: Ruling out households themselves (with two steps, they can always come back to themselves)
        - 137: Ruling out households with distance 1 (think about a triangle)

Part 6

- 142-143: Logistic regression by using leaders to estimate β

Part 7

- 147-160: Calculate the moments in the case where $q_N = q_P$
    - 153-160: Calculate moments at each grid of $q_N (= q_P)$
        - 156: Call the function `divergence_model` which calculates moments
- 163-178: Calculate moments in the case where $q_N \neq q_P$
    - 169-174: Calculate moments at each grid of (q_N, q_P)
        - 172: Call the function `divergence_model` which calculates moments
- Note: In the calculations here, a matrix D is obtained. This is a length(q_N grids)-by-length(q_P grids) matrix, and each element is a (# villages)-by-(# moments) matrix.

Part 8

- 188-194: Select if just obtaining point estimates or standard errors through bootstrap
  - bootstrap == 0: point estimates
  - bootstrap == 1: standard errors through bootstrap (1000 times)
- 197-207: Select if two step optimal weights are used or not
  - twoStepOptimal == 1:
    - 202: With some guess of $q_N(= 0.09)$ and $q_P(= 0.45)$, obtain moments by the function `divergence_model`
    - 203: Calculate the outer product of the moments, divided by the number of villages
    - 204: Take the inverse of the above expression, which will be the weight in the second step
    - **Note**: I guess usually people (i) solve the GMM problem with an identity matrix as a weight to get the consistent estimate of parameters, and (ii) get the weight in the way described above. Here, somehow authors set q_N and q_P and calculate the weight based on these parameters. If these are not derived by consistent estimates, then the weight is inconsistent and thus the estimates in the second step will be biased as well.
  - twoStepOptimal == 0: use an identity matrix as the weight in the "second" step
- 211-216: Obtain a new 4-dimension (length(q_N grids)-by-length(q_P grids)-by-(# villages)-by-(# moments)) $Dnew$ from a nested matrix D
- 223-259: Bootstrap (note: here in the bootstrap, randomly generated village weights are used, not that villages are randomly chosen with replacement)
  - 226-231: Generate bootstrap weights
    - bootstrap == 0 (ie. for point estimate): same weights for all villages
    - bootstrap == 1 (ie. for bootstrap standard errors): randomly generated weights follow exponential distribution (normalized so that the sum of weights is 1)
  - 236-244: Calculate the criterion function for each (q_N, q_P) grid
    - 240: Calculate weighted moments
    - 242: Calculate the criterion function
  - 246-254: Derive the parameter values that minimize the objective function and store the results

3 `divergence_model.m`

"This computes the deviation of the empirical moments from the simulated ones"

Arguments

- X: adjacency matrix
- Z: covariate
- Betas: estimates of β's
- leaders: vector indicating leaders
- TakeUp: vector indicating who took up microfinance

- Sec: matrix indicating second neighbors (= households with distance two)
- theta: parameter values (q_N, q_P)
- m: number of moments
- S: number of simulations
- T: number of months
- EmpRate: proportion of non-leaders who took up MF
- version: specification of which moments to use

Main part

- 23-39: Calculate empirical and simulated moments
  - 25: Calculate empirical moments, using a function `moments`
  - 28-32: Calculate simulated moments
    - 30: Simulate the take-up of microfinance, using a function `diffusion_model`
    - 31: Calculate simulated moments based on the simulated take-ups, using a function `moments`
  - 36: Take the different of empirical and simulated moments

4 moments.m

Arguments

- X: adjacency matrix
- leaders: vector indicating leaders
- infected: vector indicating who took up microfinance
- Sec: matrix indicating second neighbors (= households with distance two)
- j: village index
- version: specification of which moments to be used

Main part

- 3: Declare a variable “netstats” which contains information of networks in each village (for the function `persistent`, see here)
- 7-49: If a variable “netstats” is already defined, then almost skip the parts, but if not yet, then calculate each statistics of the networks (7-43)
  - 8: Use a function `breadthdistRAL` to calculate ??? (R: ???, D: ???)
  -
- 52-: Moments under different versions
  - 53-88: Case1 (moments (2)-(6) in the paper):
    - 54-62: “Fraction of nodes that have no taking neighbors but are takers themselves” (moment (2) in the paper)

- 56: Number of infected neighbors:
  `ones(N,1)*infected'`: a matrix with indicators of infected households in each row
  ⇒ `(ones(N,1)*infected').*X`: a matrix with indicators of infected neighbors for each household
  ⇒ `sum((ones(N,1)*infected').*X, 2)`: number of infected neighbors for each household
- 58-59: If there is any household who are linked with other households (netstats(j).degree > 0) but has no infected neighbors (infected Neighbors == 0), calculate the fraction $\frac{\text{\# HH who are linked with other households, don't have any infected neighbors, but are infected themselves}}{\text{\# HH who are linked with other households and don't have any infected neighbors}}$
- 60-61: If there is no such household in the village, then just let the moment be 0 (since the denominator of the above fraction will be 0 in this case)

- ∗ 64-69: "Fraction of individuals that are infected in the neighborhood of infected leaders stats(1) = 0" (moment (3) in the paper)
  - If `sum(netstats(j).neighborOfInfected) > 0` (???), calculate the fraction (???)
  - Otherwise, just let the moment be 0
- ∗ 71-76: "Fraction of individuals that are infected in the neighborhood of non-infected leaders)" (moment (4) in the paper)
  - If `sum(netstats(j).neighborOfNonInfected) > 0` (???), calculate the fraction (???)
  - Otherwise, just let the moment be 0
- ∗ 78-82: "Covariance of individuals taking with share of neighbors taking" (moment (5) in the paper)
  - 79: NonHermits: Indicator of non-isolated households
  - 80: ShareofTakingNeighbors: For each non-isolated household, $\frac{\text{\# infected neighbors}}{\text{\# neighbors}}$
  - 81: NonHermitTakers: Indicator of non-isolated households who took up
  - 82: moment = $\frac{\sum_{i:\text{non-isolated}} (\text{Take-up})_i \times (\text{Share of taking neighbors})_i}{\text{\# Non-isolated households}}$
- ∗ 82-88: "Covariance of individuals taking with share of second neighbors taking" (moment (6) in the paper)
  - 86: infectedSecond: number of infected second neighbors
  - 87: ShareofSecond: Share of infected second neighbors for non-isolated households, $\frac{\text{\# infected second neighbors}}{\text{\# first neighbors}}$ (** Why is the denominator about first neighbors? **)
  - 88: moment = $\frac{\sum_{i:\text{non-isolated}} (\text{Take-up})_i \times (\text{Share of taking second neighbors})_i}{\text{\# Non-isolated households}}$

- – 91-112: Case2 (moments (2), (5), and (6) in the paper):
  - ∗ 92-100: "Fraction of nodes that have no taking neighbors but are takers themselves" (moment (2) in the paper)
  - ∗ 102-106: "Covariance of individuals taking with share of neighbors taking" (moment (5) in the paper)
  - ∗ 110-112: "Covariance of individuals taking with share of second neighbors taking" (moment (6) in the paper)
- – 115-141: Case3 (moments (2), (5), and (6) in the paper, same as case 2, but purged of leader injection points):
  - ∗ 117: a variable that denotes whether a node is a leader
- – 143-169: Case4 (moments (2), (5), and (6) in the paper, same as case 2, but purged of all leader points):
  - ∗ 117: a variable that denotes whether a node is a leader
  - ∗ 166-167: Moment (6), but second neighbors who are leaders are not taken into account (see line 167)

5 `breadthdistRAL.m`

TBD...

6 `diffusion_model.m`

Arguments

- parms: parameters (q_N, q_P)
- Z: covariate
- Betas: estimates of β's
- X: adjacency matrix
- leaders: vector indicating leaders
- j: village index
- T: number of months
- EmpRate: proportion of non-leaders who took up MF

Main part

- 7-11: Prepare arrays to be updated in the simulation below
  - 7: `infected`: people who are already infected and newly infected (initial value = false for everyone)
  - 8: `infectedbefore`: people who are already infected (initial value = false for everyone)
  - 9: `contagiousbefore`: people who are already informed (initial value = false for everyone)
  - 10: `contagious`: people who are newly informed and already informed (?) (initial value = true only for leaders)
  - 11: `dynamicInfection`: vector that tracks the infection rate for the number of periods it takes place
- 16-39: Loop to simulate the diffusion for T periods
  - 20-27: Step 1: Take-up decision based on newly informed
    - 21: Probability of take-up (calculated for everyone, both non-infected and already-infected)
    - 22: Updated infected households:
      The condition `(∼contagiousbefore & contagious & x(:,t) < LOGITprob)` is satisfied if:
      - not contagious before, and
      - newly informed, and
      - the random number generated is smaller than the LOGITprob calculated in line 21.
    - 25: Update `infectedbefore`
    - 26: Update `contagiousbefore`
    - 27: `C`: number of informed households
  - 29-31: Step 2: Information flows
    - 30: `transmitPROB`: Vector of probability of transmission (individual specific, depending on informed and on take-up)
      - If informed & take-up, probability = q_P

    - If informed & not take-up, probability = q_N
  - 31: `contagionlikelihood`: The probability to information flowing from informed households to neighbors (no matter whether those neighbors are already informed or not)

    eg) Suppose that $X = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$, transmitPROB $= \begin{pmatrix} q_P \\ 0 \\ 0 \\ q_N \end{pmatrix}$, and contagious $= \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$.

    Then,

$$\begin{aligned} & \texttt{X(contagious,:).*(transmitPROB(contagious)*ones(1,N))} \\ &= \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} .* \begin{pmatrix} q_P & q_P & q_P & q_P \\ q_N & q_N & q_N & q_N \end{pmatrix} \\ &= \begin{pmatrix} 0 & q_P & q_P & q_P \\ q_N & q_N & 0 & 0 \end{pmatrix} \end{aligned}$$

- 33-36: Step 3: Simulate newly informed households
  - 34: Update `contagious`:
    - `(contagionlikelihood > rand(C,N))`: simulation of whether households are informed from already informed neighbors
    - `((contagionlikelihood > rand(C,N))'*ones(C,1) > 0)`: informed at least from one neighbor
  - 36: Update `dynamicInfection` to keep track of the fraction of take-up households