In: *Volume title,* edited by Jan Editor and Ed Janitor. BeLLS Vol. N (2017), DOI N.

Detection of COVID-19 individuals using cell-phone data

Zahra Gholamalian

Abstract. This document provides a method to help to control covid_19 pandemic, through estimating how likely one is infected with the virus. To this goal, we use an extension of HMM, in order to incorporate mobile data to the model.

1 introduction

Personal awareness of the possibility of infection with the virus, can be a self-controlling factor, decreasing the disease spread.

2 Problem formulation

Our goal is to determine the probability of infection with corona-virus, for every individual in a given city, in a specific week based on the confirmed cases in the last week among their family, and ones they had face-to-face meeting with. More specifically, let $\mathcal{V}$ be the set of *n* citizens, $X_{i,t} \in \{0, 1\}$, $i \in \mathcal{V}$, be the health state of individual *i* during week *t*, where 0 represents the healthy state, and 1 represents the infected state. Having meeting with distance up to two meters or having a person with positive COVID-19 test result in her apartment or family, increase the probability of being infected for individual *i*. Define $\mathcal{F}_i \subset \mathcal{V}$ as the family of individual *i* (those individuals that share the same house/apartment with individual *i*, such as her family), and $Y_{f_i,t}$, as the COVID-19 test result of $\mathcal{F}_i$ at *t*, which is a binary variable equal to 1, when at least one of the family members has the positive test result, and 0 otherwise. Let $\mathcal{C}_{i,t} \subset \mathcal{V}$ be the contacts of individual *i*, who are those that met face-to-face with distance up to two meters, during week *t*. Our goal is to determine the probability $P(X_{i,t}|Y_{f_i,t}, X_{\mathcal{C}_i,t-1})$; that is, the heath state of individual *i*, based on, the test result of his family members and the health state of his contacts, during the last week. In order to predict health state of individual *i* in week *t*, knowing her health state in the last week; *t-1*, is

enough and this is not necessary to know about other previous weeks, which means that given $X_{i,t-1}$, $X_{i,t}$ is independent of all previous values of *i*'s health state sequence, therefor $X_{1:T}$ is a first order Markov chain. By observing $Y_{f_i,t}$, we get information about $X_{i,t}$, so by a Hidden Markov Model (HMM)[1], we can model two stochastic processes $X_{i,t}$ and $Y_{f_i,t}$. Then to model a family, we have a number of HMMs with the common observations $Y_{f_i,t}$, we use Factorial HMM [1]; which is an extension of HMM, for the case that a number of hidden processes together cause common observations. More over, $X_{i,t}$, depends on the last hidden state of its contacts $X_{c,t-1}, \forall c \in \mathcal{C}_{i,t}$, hence to represent this dependence, we use Coupled Hidden Markov Model (CHMM)[1]; which is a model for the case that a number of hidden stochastic processes have influence on each other, and each has their one observations. Dependencies among HMMs at *t*, are determined based on $\mathcal{C}_{i,t}$, for each individual *i*, and we model this dependencies by *Noisy Or*; that means *i* is healthy if all members of $\mathcal{C}_{i,t}$ have been healthy during *t*-1,so we have:

$$P(X_{i,t} = 0) = \prod_{c \in \mathcal{C}_{i,t}} P(X_{c,t-1} = 0) \tag{1}$$

As the dependencies between HMMs follow a friendship graph we call this model GCHMM (Graph Coupled HMM).

3 Parameter Learning

Define:
$\alpha \sim P(X_{i,t} = 1 | X_{i,t-1} = 0, X_{j,t-1} = 1, \forall j \notin C_{i,t-1})$; probability of being infected from someone outside the network.
$\beta \sim P(X_{i,t} = 1 | X_{i,t-1} = 0, X_{j,t-1} = 1, \forall j \in C_{i,t-1})$; probability of being infected from someone inside the network.
$\gamma \sim P(X_{i,t} = 0) | P(X_{i,t-1} = 1)$; recovery probability if infected at previous time step,
where α, β, γ are model parameters. For transition matrix, the probability of $P(X_{i,t} = 1 | P(X_{i,t-1} = 0)$ depends on the number of *c*s, where $c \in \mathcal{C}_{i,t-1}, X_{c,t-1} = 1$, this probability increases exponentially by the number of infected neighbors:

$$\begin{cases} \gamma & X_{n,t} = 1, X_{n,t+1} = 0 \\ 1 - \gamma & X_{n,t} = 1, X_{n,t+1} = 1 \\ 1 - (1-\alpha)(1-\beta)^{C_{n,t}} & X_{n,t} = 0, X_{n,t+1} = 1 \\ (1-\alpha)(1-\beta)^{C_{n,t}} & X_{n,t} = 0, X_{n,t+1} = 0 \end{cases} \tag{2}$$

As the observation vector of the model, $Y_{f_i,t}$, is only defined for families, we need observation value of each individual, at each time, to be able to use CHMM struc-

ture, so we should introduce $Y_{i,t}$, which is the sudo observation of the individual i, at time t, as follows. For each individual i, the number of neighbors, who are from families with positive COVID-19 test result, $Y_{f_i,t}$=1, is called the number of susceptible neighbors . For each family F_i with positive COVID-19 test result, $Y_{f_i,t}$=1, we should find family members, (for example individual j), with the most number of susceptible neighbors, and set their sudo observation value, equal to 1, $Y_{j,t}$=1. Consequently, for other members of the family, (for example individual k), this value is set equal to 0, $Y_{k,t}$=0.
For the emission probability, θ_0 and θ_1 are defined as follows:

$$P(Y_{i,t} = 1|X_{i,t} = 0) \sim Bernoulli(\theta_0) \tag{3}$$

$$P(Y_{i,t} = 1|X_{i,t} = 1) \sim Bernoulli(\theta_1) \tag{4}$$

According to graph 1 and its conditional independencies, the probability of seeing an entire state sequence/matrix X is therefore as follows[2]:

$$\begin{aligned} P(X, \alpha, \beta, \gamma) &= P(\alpha)P(\beta)P(\gamma)\prod_n P(X_{n,1})\prod_{t,n} P(X_{n,t+1}|\{X_{n'\in N,t}\}, \alpha, \beta, \gamma) \\ P(X) &= \prod_n P(X_{n,1})\prod_{t,n} \gamma^{1_{X_{n,t}=1}.1_{X_{n,t+1}=0}}.(1-\gamma)^{1_{X_{n,t}=1}.1_{X_{n,t+1}=1}}. \\ &\quad (1-(1-\alpha)(1-\beta)^{C_{n,t}})^{1_{X_{n,t}=0}.1_{X_{n,t+1}=1}}. \end{aligned} \tag{5}$$

$$((1-\alpha)(1-\beta)^{C_{n,t}})^{1_{X_{n,t}=0}.1_{X_{n,t+1}=0}} \tag{6}$$

4 Inference

In many models with hidden variables, Maximum Likelihood Estimate (MLE) for parameter estimation does not work, duo to marginalization of hidden variables. In such cases, it is often an appropriate choice, to use an algorithm called expectation maximization, or EM for short (Dempster et al. 1977; Meng and van Dyk 1997; McLachlan and Krishnan 1997). This is a simple iterative algorithm, often with closed-form updates at each step. EM exploits the fact that if the data were fully observed, then the ML/ MAP estimate would be easy to compute. EM is an iterative algorithm which alternates between inferring the missing values given the parameters (E step), and then optimizing the parameters given the "filled in" data (M step)[1].
In our model, Y_i and X_i are the observation and hidden state matrix for the i'th dataset (if we have N datasets), respectively. The goal is to maximize the log likelihood of the observed data:

$$l(\theta) = \sum_{i=1}^{N} log\ p(Y_i|\theta) = \sum_{i=1}^{N} log\ \left[\sum_{X_i} p(Y_i, X_i|\theta)\right] \tag{7}$$

Unfortunately this is hard to optimize, duo to the time complexity, since the log cannot be pushed inside the sum. Consider an arbitrary distribution $q(X_i)$ over the hidden variables. The observed data log likelihood can be written as follows:

$$l(\theta) = \sum_{i=1}^{N} log\ \left[\sum_{X_i} p(Y_i, X_i|\theta)\right] = \sum_{i=1}^{N} log\ \left[\sum_{X_i} q(X_i)\frac{p(Y_i, X_i|\theta)}{q(X_i)}\right] \tag{8}$$

Since log(·) is a concave function, from Jensen's inequality we have the following lower bound:

$$l(\theta) \geq \sum_{i}\sum_{X_i} q(X_i)\frac{P(Y_i, X_i|\theta)}{q(X_i)} \tag{9}$$

The right side of equation (9) can be rewritten as:

$$\sum_{i}\left[\sum_{X_i} q(X_i) log\ P(Y_i, X_i|\theta) + \sum_{X_i} -q(X_i) log\ q(X_i)\right] \tag{10}$$

The first term in the above equation is expectation of $P(Y_i, X_i|\theta)$, under the probability distribution q_i, and the last term is equal to the entropy of q_i. Now let us denote the lower bound as follows:

$$Q(\theta, q^t) \triangleq \sum_{i} \mathbb{E}_{q_i^t}\left[log\ P(Y_i, X_i|\theta)\right] + \mathbb{H}(q_i^t) \tag{11}$$

The above argument holds for any positive distribution q. It can be shown that, setting $q_i(X_i) = P(X_i|Y_i, \theta)$, yields the tightest lower bound[1]. By choosing posterior probability distribution as q_i, $Q(\theta, q^t)$ is called expected complete data log likelihood.

$$Q(\theta, q^t) \triangleq \sum_{i} \mathbb{E}_{P(X_i|Y_i,\theta)}\left[log\ P(Y_i, X_i|\theta)\right] + \mathbb{H}(q_i^t) \tag{12}$$

We use $P(X_i, Y_i) = P(Y_i|X_i).P(X_i)$, in equation (11) to compute $Q(\theta)$ for our model, the second term in (11) is constant with respect to θ, so it can be omitted.

$$Q(\theta, q^t) \triangleq \sum_{i} \mathbb{E}_{P(X_i|Y_i,\theta)}\left[log\ P(Y_i|X_i, \theta) + logP(X_i)\right] \tag{13}$$

The first term in the expectation in (13), is due to emission probability, and the last term is the probability of seeing the entire state sequence, calculated in (6). One way to calculate this expectation is through Monte Carlo integration. The Monte Carlo integration method, estimates the expectation of a function $\phi(x)$ under a probability distribution f(x), by taking samples $\{x^{(j)}\}_{j=1}^{J} : x^{(j)} \sim f(x)$. An unbiased estimate, $\hat{\Phi}$, of the expectation of $\phi(x)$ under f(x), using J samples is given by:

$$\Phi = \int dx\ f(x)\phi(x) \simeq \hat{\Phi} = \frac{1}{J}\sum_{j=1}^{J}\phi(x^{(j)}). \tag{14}$$

By choosing posterior distribution $P(X_i|Y_i,\theta)$ as $f(x)$, and joint distribution $P(Y_i, X_i|\theta)$ as $\phi(x)$, Monte Calro method helps to find a good estimate for expected complete data log likelihood; $Q(\theta, q^t)$. The index i is for the dataset, for $i = 1, 2, ..., N$, from now we omit this index, assuming that we have just one dataset, for the sake of simplicity. More precisely, in the E step, by randomly initializing parameters, expectation is computed by sampling $\{x^{(j)}\}_{j=1}^{J}$ from $P(X|Y,\theta)$, and averaging over $P(Y, X^{(j)}|\theta)$ for $j = 1, 2, ..., J$. So we need to sample from posterior distribution, and then average over the joint distribution to approximate $Q(\theta)$. Gibbs sampling is an iterative algorithm which estimates posterior distribution. This method, starts out by generating a sample of the unobserved variables from some initial distribution; Starting from that sample, we then iterate over each of the unobserved variables, sampling a new value for each variable given our current sample for all other variables. This process allows information to "flow" across the network as we sample each variable[3]. Having the sudo observations $Y_{i,t}$ for $i = 1, 2, ..., n$, and $t = 1, 2, .., T$, applying this algorithm to our model, will generate samples of posterior probability distribution, over the variables $X_{i,t}$ for $i = 1, 2, ..., n$, and $t = 1, 2, .., T$. Our Gibbs Sampling algorithm begins by generating one sample, say by forward sampling. In the first iteration, it would now resample all of the unobserved variables, one at a time. Thus, assume that we want to sample $X_{1,t}$ from the distribution $P(X_{1,t}|\{X, Y\} \setminus X_{1,t})$. Note that because we are computing the distribution over a single variable given all the others, this computation can be performed very efficiently:

$$P(X_{1,t}|\{X, Y\} \setminus X_{1,t}) = \frac{P(X, Y)}{\sum_{X_{1,t}=0,1} P(X, Y)} \tag{15}$$

For the graph shown in figure 2, joint distribution probability is as follows:

$$\begin{aligned}P(X,Y) =& P(X_{1,t-1})P(X_{2,t-1})P(X_{3,t-1})P(X_{1,t}|X_{1,t-1},X_{3,t-1})P(X_{2,t}|X_{2,t-1}).\\ & P(X_{3,t}|X_{1,t-1},X_{3,t-1})P(X_{1,t+1}|X_{1,t},X_{3,t})P(X_{2,t+1}|X_{2,t},X_{3,t}).\\ & P(X_{3,t+1}|X_{1,t},X_{2,t},X_{3,t})P(Y_{1,t-1}|X_{1,t-1})P(Y_{2,t-1}|X_{2,t-1}).\\ & P(Y_{3,t-1}|X_{3,t-1})P(Y_{1,t}|X_{1,t})P(Y_{2,t}|X_{2,t})P(Y_{3,t}|X_{3,t}).\\ & P(Y_{1,t+1}|X_{1,t+1})P(Y_{2,t+1}|X_{2,t+1})P(Y_{3,t+1}|X_{3,t+1})\end{aligned} \tag{16}$$

If we apply (15) in (14), only terms including $X_{1,t}$ will remain in the fraction. So we have:

$$P(X_{1,t}|\{X,Y\}\setminus X_{1,t}) = \frac{P(X_{1,t}|X_{1,t-1},X_{3,t-1})P(X_{1,t+1}|X_{1,t},X_{3,t})P(X_{3,t+1}|X_{1,t},X_{2,t},X_{3,t})P(Y_{1,t}|X_{1,t})}{\sum_{X_{1,t}=0,1}P(X_{1,t}|X_{1,t-1},X_{3,t-1})P(X_{1,t+1}|X_{1,t},X_{3,t})P(X_{3,t+1}|X_{1,t},X_{2,t},X_{3,t})P(Y_{1,t}|X_{1,t})} \tag{17}$$

By use of (16), for $X_{1,t} = 0\ and\ 1$, two parameters λ_0, λ_1, can be obtained respectively, where:

$$\lambda_0 = \frac{P(X_{1,t}=0|\{X,Y\}\setminus X_{1,t})}{(P(X_{1,t}=0|\{X,Y\}\setminus X_{1,t})+P(X_{1,t}=1|\{X,Y\}\setminus X_{1,t}))} \tag{18}$$

$$\lambda_1 = \frac{P(X_{1,t}=1|\{X,Y\}\setminus X_{1,t})}{(P(X_{1,t}=0|\{X,Y\}\setminus X_{1,t})+P(X_{1,t}=1|\{X,Y\}\setminus X_{1,t}))} \tag{19}$$

So sampling new value for $X_{1,t}$, is the same as sampling from a binomial distribution with success parameter λ_1. This process can be done quite simply as follows: We generate a sample d uniformly from the interval [0, 1). We then partition the interval into 2 subintervals: [0, λ_0), [λ_0, 1); as λ_0+λ_1=1. If d is in the first interval, then the sampled value is equal to 0, otherwise 1. Having sampled $X_{1,t}$, we now continue to resample $X_{i,t}$ for $i = 1, ...n$ and $t = 1, ...T$ one at a time; note that the distribution is conditioned on the newly sampled value $X_{1,t}$ and other previous variables. The process now repeats K times and the distribution from which we generate each sample, gets closer and closer to the posterior. We run Gibbs sampling J times, to take J samples $\{x^{(j)}\}_{j=1}^{J} : x^{(j)} \sim P(X|Y,\theta)$. As $P(X^{(j)},Y) = P(Y|X^{(j)}).P(X^{(j)})$ for $j = 1, 2, ..., J$, joint distribution and $Q(\theta)$ can be computed from equations (3),(4),(6) and (11). This process in brief
E step:
1- Randomly initialize parameters: $\alpha,\ \beta, \gamma, \theta_0,\ \theta_1$
2- Run Gibbs Sampling J times to sample J vectors $(X^{(j)}, Y)$ for $j = 1, 2, ..., J$, for each $i = 1, 2, ..., N$
3- Use Monte Carlo method (6), to estimate expected complete data log likelihood

ID	result	Day
1	+	20/5/2020
16	-	18/6/2020
72	-	18/6/2020
40	+	13/7/2020
15	-	14/7/2020

Table 1: Test results

as:

$$Q(\theta) = \frac{1}{J}\sum_{j=1}^{J} log\ P(Y|X^{(j)},\theta)P(X^{(j)}) \tag{20}$$

In the M step, by maximizing $Q(\theta)$, model parameters are updated. For each parameter, updated value is the result of setting the gradient of $Q(\theta)$ with respect to that parameter, equal to zero.

5 Data

Consider the case that person *i*'s test result is positive. This will be registered in the hospital or laboratory with a mobile phone number, which we assume that belongs to person *i* or anyone in her family $\mathcal{F}_i$ (There is a small probability that the number belongs to a person out of $\mathcal{F}_i$, which we ignore). If we give an ID to each mobile phone number, e.g, p_i then we have a table of ID's and their test results, for each day *d*, which we call Table 1. On the other hand, a telecommunication operator has *m* BTS towers. When a call is made, tower b_j records two phone IDs, p_i, p_j and exact time t, so we have a table of p_i, p_j, t, $date$ for b_j, which we call Table 2. Assume that for each b_j, for a month interval, we draw the social network of Table 2, and we call it G_j, where the nodes are p_is, and an edge between two nodes represents the call that have been recorded between them in the Table 2. By running community detection algorithm on G_j, we can find out family clusters in the domain of each BTS, and determine $\mathcal{F}_i$. Additionally, if for each day *d* in week *t*, we look at Table 2, each pair p_i, p_j, that have a large number of calls during *d*, we assume that they met on day *d*, and hence add individual *j* to the contact list C_i.

From Table 1, for each day $d \in t$ (week *t*), if *j* has positive COVID-19 test result and $j \in F_i$, then $Y_{f_i,t}$=1.

ID_1	ID_2	time	date
1	117	10:32:33	20/5/2020
16	56	17:42:03	20/5/2020
72	4	20:02:58	20/5/2020
40	309	6:10:13	23/6/2020
15	20	11:32:33	23/6/2020

Table 2: BTS call log example

Figure 1: GCHMM

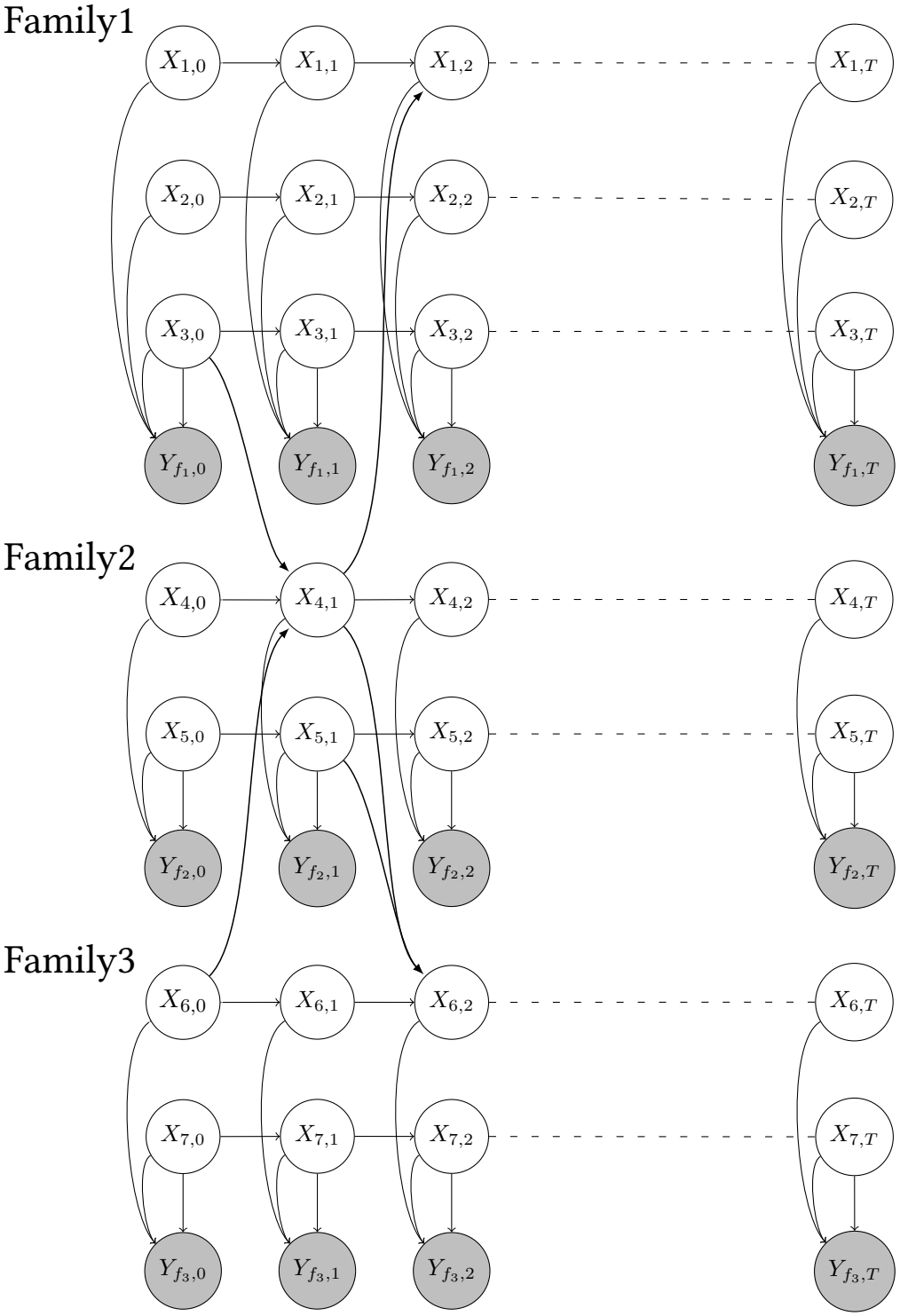

Figure 2: An example network to explain Gibbs sampling algorithm