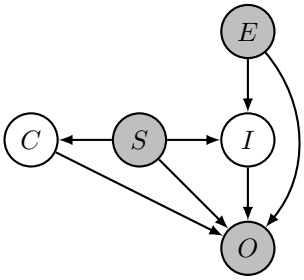
COVID-19 Tracing
Variational inference & training

May 2020

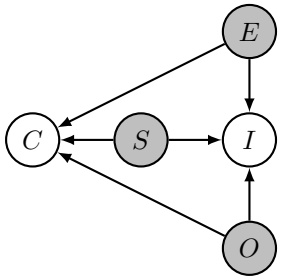
Contents

1 Introduction

Instead of creating a generative model over the whole population to model exactly the transmission of the infection (which would require the whole contact graph), we want to define the generative and inference models at the level of an *individual phone*. All the information about the encounters contained in the messages received by the phone serve as observations in our models, and are sufficient to define all the necessary conditional probability distributions; they d-separate what happens on Alice's phone from what happens on other phones.



(a) Generative model
$p(I, C, O \mid S, E)$



(b) Inference model
$q(I, C \mid O, S, E)$

Figure 1: Generative & inference models, at the level of an individual phone. Empty nodes are latent variables, and shaded nodes are observations. Notations are given in Table 1.

The exact structures of the generative model and the inference model are shown in Figure 1, with the notations and definitions of the variables in Table 1. In Section 2 we provide details about the generative model $p(I, C, O \mid S, E)$, in Section 3 we detail how the inference model $q(I, C \mid O, S, E)$ is defined, and finally in Section 4 we show how we can train both models using data.

1.1 Why not a Dynamic Bayesian Network?

This model is not a Dynamic Bayesian Network; the variables E, I, and O contain all the information from the past 14 days, and do not depend *explicitly* (i.e. in the structure of the network) on the corresponding variables at the previous timestep. There are multiple reasons why we might not want to use a Dynamic Bayesian Network as the generative model

1. The temporal aspect required by the (asynchronous) Dynamic Bayesian Network might not be guaranteed. The observations about the encounters in E is only a *set*, and the order of the encounters might not be preserved. Even if we use a time resolution of a day for the DBN, we might receive updated risks from multiple days in the past that would not be handled properly by a DBN.

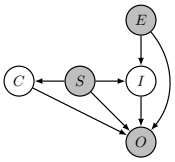
Variable	Type	Description
S	– $(*)$	**Static information** about the individual (e.g. demographic, pre-existing medical conditions).
E	– $(*)$	Set of **encounter observations** for all encounters in the past 14 days (e.g. risk level, approximate repeated encounter, duration).
O	Binary vector	List of all the **symptoms & test results** for the past 14 days.
C	Continuous vector	The **contagiousness profile** (i.e. contagiousness as a function of time). This is a static property of the individual, see Section 2.1.1.
I	Binary vector $(**)$	The **infection events**, i.e. which encounters were responsible of the infection.

Table 1: Notations used throughout the document. $(*)$ The variables S and E are not modeled in the generative model (see Section 2), therefore their exact type is non-essential. $(**)$ The variable I is a binary vector in the generative model p, but is a Categorical variable in the inference model q.

2. The information contained in E might not be consistent from one day to the next. For example, since we are running the clustering algorithm every day on the phone, and the result is sent as part of E, one contact might end up in different clusters from one day to another. If we can't ensure that information in common between two timesteps (i.e. intersection of the messages in E yesterday and E today), we are losing the advantage of using DBNs.

3. Defining the conditional probability distributions for the generative model in Figure 1a is simple enough and follows nicely what is currently implemented in the simulator. While we might be losing some temporal consistency that the DBN would provide, this model can constitute a first version of the generative model.

2 Generative model



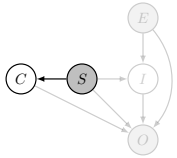
The generative model $p(I, C, O \mid S, E)$, shown in Figure 1a, is a model conditioned on both the static information S and encounters E defined by

$$p(I, C, O \mid S, E) = p(C \mid S)\, p(I \mid S, E)\, p(O \mid I, C, S, E). \tag{1}$$

One reason why we choose to define the generative model as a conditional model is that we consider S and E as observed variables. It would be easy enough to add both a *demographics model* $p(S)$ (e.g. changing from one country to another) and a *mobility model* $p(E \mid S)$ on top.

In the following sections, we detail how these different conditional probability distributions can be parametrized, both using the parametrization of the current simulator, as well as a parametric model that can eventually be learned (e.g. using neural networks).

2.1 Probability of contagiousness profile



The latent variable C represents the *contagiousness profile*, that is the *viral load* as a function of time. From this latent variable, it is possible to derive the contagiousness level using I (e.g. the number of days since infection).

2.1.1 In the simulator

Currently in the simulator, the contagiousness profile is a piecewise linear function that depends on 5 parameters: the *infectiousness onset*, the *plateau start*, the *plateau height*, the *plateau duration*, and the *recovery duration*. See Figure 2 for an illustration.

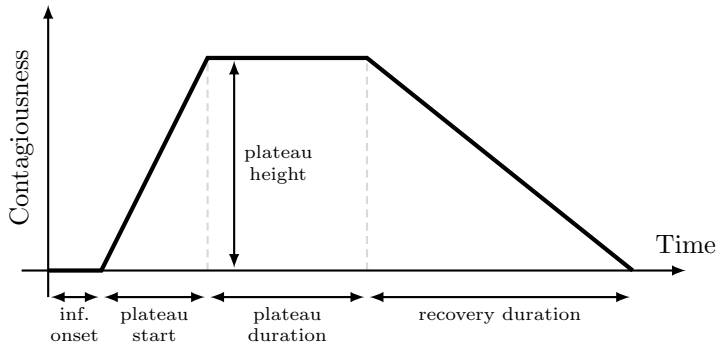


Figure 2: Contagiousness profile. This piecewise linear function represents the evolution of the contagiousness with time. Time $t = 0$ corresponds to exposure.

Here, the contagiousness profile is then the equivalent to these 5 parameters

$$C = \{\text{infectiousness onset}, \text{plateau start}, \text{plateau height}, \\ \text{plateau duration}, \text{recovery duration}\}.$$

In the simulator, these parameters are specific to an individual and are generated

according to the following process:

$$
\begin{aligned}
p(\text{infectiousness onset}) &= \mathcal{N}(1 + \mu_{io}, \sigma_{io}^2) && \text{code} \\
p(\text{plateau start}) &= \bar{\mathcal{N}}(\mu_{ps}, \sigma_{ps}^2) && \text{code} \\
p(\text{plateau duration}) &= \bar{\mathcal{N}}(\mu_{pd}, \sigma_{pd}^2) && \text{code} \\
p(\text{plateau height} \mid \text{age}) &= \mathcal{U}\left(\frac{\text{age}}{200} + a_{ph}, \frac{\text{age}}{200} + b_{ph}\right) && \text{code} \\
p(\text{recovery duration} \mid \text{age}) &= \bar{\mathcal{N}}\left(\frac{\text{age}}{10} + \mu_{rd} - 1, \sigma_{rd}^2\right), && \text{code} \qquad (2)
\end{aligned}
$$

where $\bar{\mathcal{N}}$ is a truncated Normal distribution, and $\mathcal{U}$ is the uniform distribution. This defines $p(C \mid S)$, and the parameters of these distributions are fixed for now. Note that the contagiousness profile currently only depends on the age of the individual.

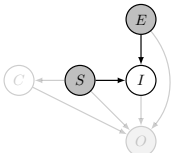
2.1.2 Modeling

To allow more flexibility, we can parametrize the different distributions defined in Equation 2 by neural networks, with all the static information S about the individual. This would allow us to have more control over the way the contagiousness profile is defined, and could be interesting for epidemiologists; for example, how does the contagiousness profile change with age, sex, or pre-existing conditions. Moreover to enforce having positive quantities, we can use log-Normal distributions instead of truncated Normal distributions. If we call θ all the parameters of the different neural networks, this becomes:

$$
\begin{aligned}
p(\text{infectiousness onset} \mid S) &= \log\mathcal{N}(\mu_{io}(S;\theta), \sigma_{io}^2(S;\theta)) \\
p(\text{plateau start} \mid S) &= \log\mathcal{N}(\mu_{ps}(S;\theta), \sigma_{ps}^2(S;\theta)) \\
p(\text{plateau duration} \mid S) &= \log\mathcal{N}(\mu_{pd}(S;\theta), \sigma_{pd}^2(S;\theta)) \\
p(\text{plateau height} \mid S) &= \mathcal{U}(a_{ph}(S;\theta), b_{ph}(S;\theta)) \\
p(\text{recovery duration} \mid S) &= \log\mathcal{N}(\mu_{rd}(S;\theta), \sigma_{rd}^2(S;\theta)), \qquad (3)
\end{aligned}
$$

where $\log\mathcal{N}$ is a log-Normal distribution. Even though we could have a general parametrization (not necessarily piecewise linear) of the contagiousness profile here, we are using domain knowledge from epidemiologists to impose structure[1].

2.2 Probability of infection



The latent variable I is a vector of binary random variables that represents when the individual got infected (and if they even got infected). From this latent variable, it is possible to derive the number of days since infection. The

[1] Alternatively, we could parametrize C as a vector of 15 continuous values, one for the viral load of each day. We would lose the structure (rising and falling edge) and interpretability though.

vector I contains $n + 14$ elements, where n is the number of contacts in the past 14 days (i.e. the size of E), the "+ 14" represents unmeasured infections in the past 14 days. Note that being not infected corresponds to I being the zero vector.

The observation of encounters E contains information about the different contacts in the past 14 days. In particular, E contains for all contacts:

1. The contagiousness level of the contact. Note that we don't observe the true contagiousness level here, but only a quantized version of the contagiousness level (this is the *risk level*).
2. An approximate value of whether or not the contact is a repeated contact, using the clustering algorithm on the phone.
3. An approximate value of the distance and duration of the encounter.
4. Eventually, this will also contain coarse information about the location

2.2.1 In the simulator

Let I_t, with $t \leq n$, be the binary random variable representing the event that the encounter E_t was the encounter that infected the individual. In the simulator, the probability of I_t is defined by

$$p(I_t = 1 \mid E_t, S) = \begin{cases} p & \text{if contact close enough \& long enough} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The decision if the contact is close enough and long enough can be made using the information of distance and duration of the encounter contained in E_t (approximate value); the thresholds are handcrafted. The probability of infection p also depends on information from E_t, and is defined by

$$p = \text{contagiousness of contact} \times \text{proximity factor}.$$

There is also a probability of infection from the environment, which can be modeled once we have access to coarse location information in E_t. Unmeasured infection is handled in the simulator through contacts with people who might not have the app, however there is no simple $p(I_t \mid S)$ for $t > n$.

2.2.2 Modeling

We can replace the probability of infection by the output of a neural network, which depends on all the observed information about the encounter E_t and the static information S.

$$\begin{aligned} p(I_t = 1 \mid E_t, S) &= \sigma(f(E_t, S; \theta)) && \text{if } t \leq n \\ p(I_t = 1 \mid S) &= \sigma(g(S; \theta)) && \text{if } t > n \text{ (i.e. unmeasured infection)} \end{aligned} \tag{5}$$

where f and g are neural networks with parameters θ. Contrary to inference, the probability of infection I_t only depends on E_t (and S) in the generative model, and not the full E; in other words, no need to use a transformer in the generative model. Moreover, we also assume that the unmeasured infections I_t for $t > n$ are conditionally independent from all other I_t. The full conditional probability is therefore

$$p(I \mid S, E) = \underbrace{\prod_{t=1}^{n} p(I_t \mid E_t, S)}_{\text{measured}} \underbrace{\prod_{k=1}^{14} p(I_{n+k} \mid S)}_{\text{unmeasured}}. \tag{6}$$

Variable	Type	Description
$\bar{I}$	Binary	The event **"is infected?"**, where $\bar{I} = 0$ means that the individual was not infected.
D	Categorical	The **number of days since infection**, taking values $d \in [1, 14]$, conditioned on the individual being infected ($\bar{I} = 1$).

Table 2: Notations for some useful auxiliary variables that can be derived from I and E.

From this conditional probability distribution, we can derive some useful distributions, such as the probability of not being infected

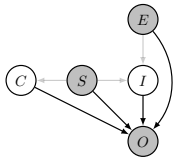
$$\begin{aligned} p(\bar{I} = 0 \mid S, E) &= p(I = \mathbf{0} \mid S, E) \\ &= \prod_{t=1}^{n} p(I_t = 0 \mid E_t, S) \prod_{k=1}^{14} p(I_{n+k} = 0 \mid S), \end{aligned} \tag{7}$$

where $\bar{I}$ is the binary variable "is infected?", as well as the probability of the day of infection $d \in [1, 14]$, given that the individual was infected (assuming a uniform prior over the days)

$$\begin{aligned} &p(D = d \mid \bar{I} = 1, S, E) \propto \\ &\quad 1 - p(I_{n+d} = 0 \mid S) \prod_{t=1}^{n} p(I_t = 0 \mid E_t, S)^{\mathbf{1}(E_t \text{ on day } d)}. \end{aligned} \tag{8}$$

2.3 Probability of symptoms & test results

The observed variable O is a vector of many binary[2] variables representing the onset of different symptoms (e.g. cough, fever) and test results over the past 14 days. For the generative model, we need to define $p(O \mid I, C, S, E)$.



[2]Should be ternary variables, to handle "not available".

Why does O depend on E? As we'll see in the following sections, it is more convenient to define the probability of symptoms & test results conditioned on the event of being infected $\bar{I}$ and the number of days since infection D (if the individual was infected). While the variable I contains almost all the information required to compute $\bar{I}$ and D, a priori I does not have any notion of "time"; I is only a set of binary events, the order of which might be arbitrary. To do the mapping between the elements of I and the corresponding days, we need the information from E. The dependency of O on E is then only purely technical, and can be safely ignored – it has no impact on the generative model.

2.3.1 In the simulator

Symptoms The different symptoms modeled at the moment in the simulator are shown in Figure 3, together with their dependencies.

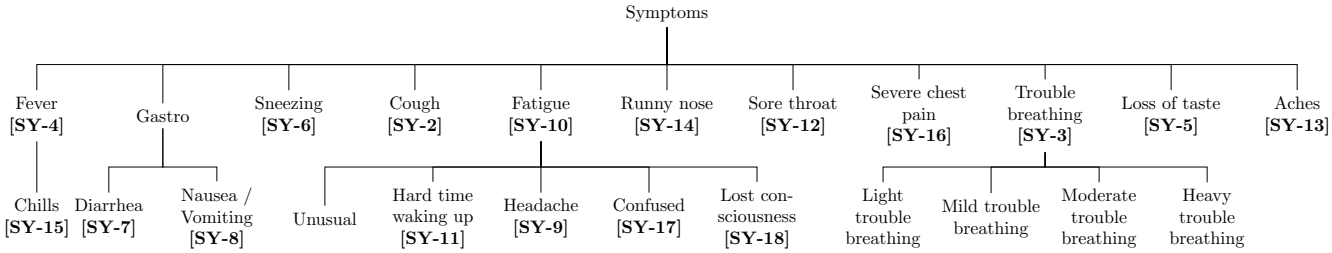


Figure 3: List of the symptoms in the simulator, with the corresponding feature IDs from the feature list document.

The symptoms and their severity are generated depending on the progression of the disease. The infection is decomposed in 6 different phases, which are shown in Figure 4: the *pre-symptomatic period*, prior to any symptom onsets, and phases from 1 to 5, which dictate the severity of the symptoms. The symptoms also show some progression, so that the symptoms at day d might depend on those at $d-1$ (e.g. the same symptom, but with a different severity).

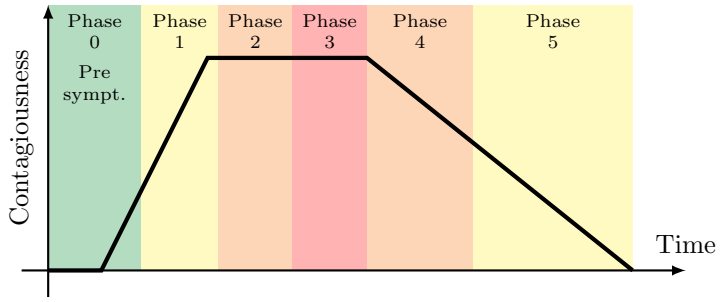


Figure 4: Progression of the symptoms in phases, with the contagiousness profile superimposed. The color represents the severity of the phase.

Without detailing its full computation, the probability of the symptoms has the following form

$$p(O \mid \bar{I}, D, C, S) = p(O_1 \mid \bar{I}, D, C, S) \prod_{d=2}^{15} p_d(O_d \mid O_{1:d-1}, \bar{I}, D, C, S), \quad (9)$$

where the different conditional probability distributions generally depend on the age and pre-existing conditions (from the static information S), the contagiousness profile C for the definition of the phases, and the number of days since infection D. Recall that $\bar{I}$ is the binary random variable "is infected?"; we have seen in Section 2.2.2 that it is possible to derive the variables $\bar{I}$ and D from I.

Test results The test result depends on whether the individual is infected or not, and a probability of false positive, based on the nature of the test.

$$p(\text{test result} = 1 \mid \bar{I}) = \begin{cases} 1 - p & \text{if the individual is infected } (\bar{I} = 1) \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where $p = 0.1$ is a fixed value for the unique test available in the simulator at the moment (lab test).

2.3.2 Modeling

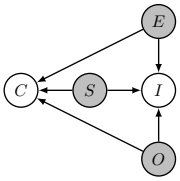
Since we want to model the progression day-by-day of the symptoms, it is more natural to define the conditional probability distribution on the number of days D since infection (and if the individual was infected $\bar{I}$), similar to the simulator, instead of directly $p(O \mid I, C, S, E)$.

Symptoms Following the way symptoms are produced in the simulator, we can go even further and define the probability of having some symptom O not on the number of days D since infection, but on the phase F. In other words, we need to model both

$$\underbrace{p(O \mid \bar{I} = 0, S, E)} \quad \text{and} \quad \underbrace{p(O \mid F = f, \bar{I} = 1, S, E)}$$

TODO: More details about how to model the symptoms.

3 Inference model



The inference model $q(I, C \mid O, S, E)$, shown in Figure 1b, is the model responsible for making approximate inference (on the phone), based on observations. This is defined by

$$q(I, C \mid O, S, E) = q(C \mid O, S, E)\, q(I \mid O, S, E). \quad (11)$$

In the following sections, we detail how these two conditional probability distributions can be represented as parametric models that can be eventually learned (e.g. using neural networks).

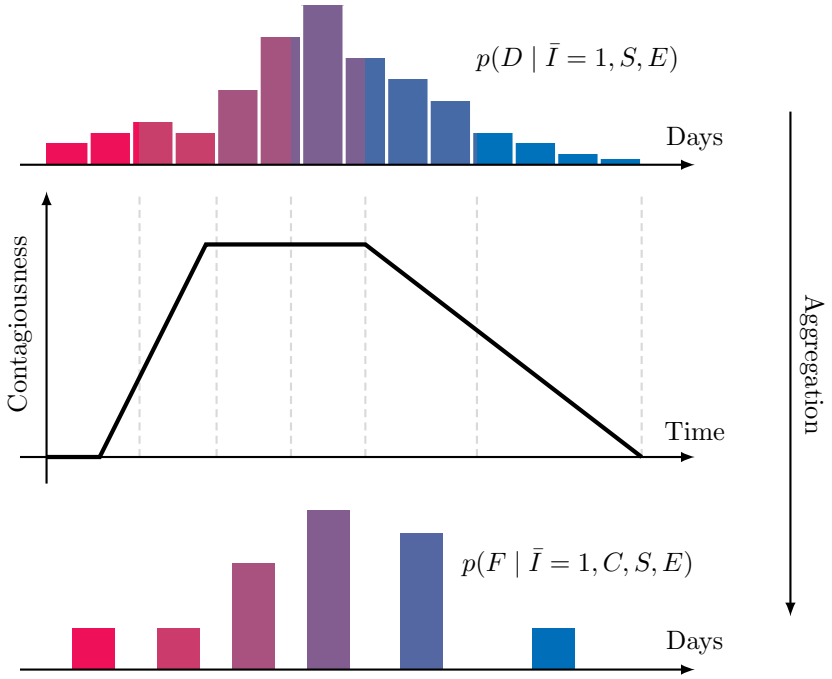
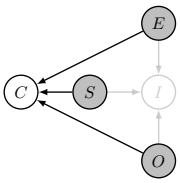


Figure 5: Probability distribution of being in phase F. This is a categorical distribution over the 6 possible phases of the disease. The probabilities are computed by aggregating the probabilities for all days in the phase.

3.1 Embedding of the observations

To encode the different observations available for the inference model, we first embed O and S using 2 separate embedding functions. The encounters E are also embedded, with different embedding functions for time (day of encounter), duration, approximate repeated encounters, and risk levels. All these embeddings are then concatenated into one large embedding matrix X of size $n \times$ embedding size.

3.2 Inference on contagiousness profile



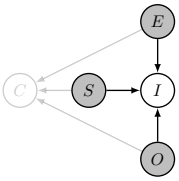
Since the variable E, and therefore the embedding X, have variable length, we need to use a neural network architecture that can handle variable-length input. One option is to use a Transformer. Similar to the way we modeled $p(C \mid S)$ in Section 2.1.2, we can take inspiration from Equation 3 and parametrize the inference model for the contagiousness profile $q(C \mid O, S, E)$ as separate distributions parametrized with Transformers taking as input the embedding matrix X defined in Section 3.1, and returning scalar values for the mean and

variance of the different distributions.

$$
\begin{aligned}
q(\text{infectiousness onset} \mid O, S, E) &= \log \mathcal{N}(\mu_{io}(X;\phi), \sigma^2_{io}(X;\phi)) \\
q(\text{plateau start} \mid O, S, E) &= \log \mathcal{N}(\mu_{ps}(X;\phi), \sigma^2_{ps}(X;\phi)) \\
q(\text{plateau duration} \mid O, S, E) &= \log \mathcal{N}(\mu_{pd}(X;\phi), \sigma^2_{pd}(X;\phi)) \\
q(\text{plateau height} \mid O, S, E) &= \mathcal{U}(a_{ph}(X;\phi), b_{ph}(X;\phi)) \\
q(\text{recovery duration} \mid O, S, E) &= \log \mathcal{N}(\mu_{rd}(X;\phi), \sigma^2_{rd}(X;\phi)). \qquad (12)
\end{aligned}
$$

In practice, we can have a single Transformer returning a vector of size 5×2 for all the parameters at once, to leverage parameter sharing; the Transformer can also be shared with the one in Section 3.3. In order to send the risk level of an individual to the contacts, we need to compute the value of the *contagiousness*. We will see in Section 3.4 that it is possible to compute this value based on both C and I inferred by the inference model q.

3.3 Inference on the infection



Like in Section 3.2, we use a Transformer architecture to handle the variable-length input X. More precisely, we can use a Set Transformer, which is an architecture that is capable of processing sets of objects. The output of the Set Transformer is a vector of size $n + 15$, where n is the number of messages (length of E), with an additional 14 values for the unmeasured infections (one for each of the past 14 days, see Section 2.2), and 1 value if there was no infection.

Using the embedding X, we can define $q(I \mid O, S, E)$ as a Categorical distribution over the output of the Set Transformer. Note that we are using a Categorical distribution here instead of a vector Bernoulli variables (as in the generative model) to simplify training; see Section 4.1.1.

$$
q(I \mid O, S, E) = \text{Categorical}\Big(\underbrace{\text{softmax}(\text{SAB}(X;\phi))}_{n+15 \text{ values}}\Big), \qquad \text{code} \qquad (13)
$$

where SAB corresponds to the Set Transformer architecture, with parameters ϕ. Note that the parameters ϕ should also include the parameters of the embedding functions. By convention, index 0 corresponds to no infection, indices $t \in [1, n]$ to measured infections with a corresponding E_t, and $t > n$ to unmeasured infections, to match the notations in Section 2.2.

Similar to Section 2.2.2, we can also derive useful distributions on the event of being infected $\bar{I}$ and the number of days since infection D (if there was infection); see Table 2 for reference. Since $q(I \mid O, S, E)$ is a Categorical distribution, this derivation is even simpler, with the probability of not being infected being

$$
q(\bar{I} = 0 \mid O, S, E) = q(I = 0 \mid O, S, E), \qquad (14)
$$

and the probability of the day of infection $d \in [1, 14]$, given that the individual was infected

$$q(D = d \mid \bar{I} = 1, O, S, E) = q(I = n + d \mid O, S, E) + \sum_{t=1}^{n} q(I = t \mid O, S, E) \cdot \mathbf{1}(E_t \text{ on day } d). \quad (15)$$

WIP: Furthermore, the probability of being infected by an unmeasured infection on day d is proportional to the probability of being infected by one of the encounters on day d, as well as the percentage of the population that is providing information in the shape of S, E, O (the adoption rate p_a).

$$q(I = n + d \mid O, S, E) \propto \quad p_a \sum_{t=1}^{n} q(I = t \mid O, S, E) \cdot \mathbf{1}(E_t \text{ on day } d).$$

However, if the sample of the population that uses the app is not randomly selected from the population, then the proportionality relationship above will have to include some form of sample selection bias correction, since the probability of being infected from a user inside the app might be different from a user outside the app, even without interventions.

3.4 Inference on contagiousness

While we are doing inference on the whole contagiousness profile (via the 5 parameters of the piecewise linear function), we need to also be able to perform inference on the value of the *contagiousness* to compute the risk level (which is a quantized version of the contagiousness). This risk level is later sent to other users. From both C and I (via $\bar{I}$ and D), it is possible to compute this value of the contagiousness $c(I, C)$.

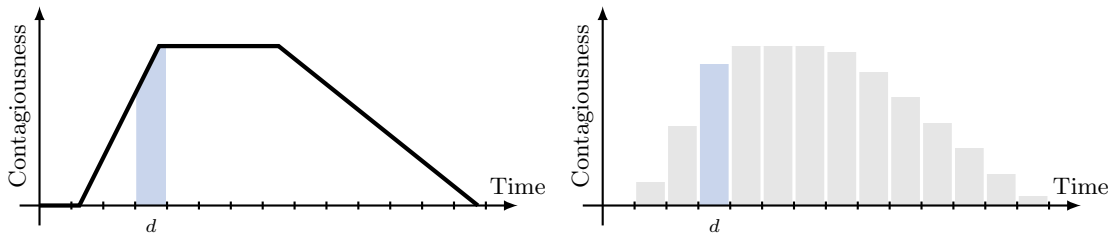


Figure 6: Computation of the average contagiousness value per day $\bar{c}_d(C)$ from the contagiousness profile C.

Figure 6 shows how to compute the average contagiousness value per day from the contagiousness profile C. For each day d, we then have the corresponding value of the contagiousness $\bar{c}_d(C)$ (Figure 6, right). To get today's contagiousness value, we can simply take the average value of the contagiousness based on

the number of days D since infection (if the individual was infected); recall that the value of D can be inferred from I and E, see Section 3.3.

$$c(I, C) = \begin{cases} 0 & \text{if } \bar{I} = 0 \text{ (i.e. no infection)} \\ \bar{c}_D(C) & \text{otherwise} \end{cases} \tag{16}$$

3.4.1 Risk level & average contagiousness

To produce a risk level for today, we can simply take a sample I and C from $q(I, C \mid O, S, E)$, and compute the contagiousness value based on Equation 16. However this value might have high variance, depending on the inference model. To mitigate this variance issue, we can instead compute the risk level based on the *average contagiousness* over I and C:

$$\text{Risk level} = \text{quantize}\big(\underbrace{\mathbb{E}_{q(I,C|O,S,E)}[c(I, C)]}_{\text{Equation 20}}\big). \tag{17}$$

Average over I To compute the expectation over the infectiousness I, we can use the distribution over D from Equation 15:

$$\mathbb{E}_{q(I|O,S,E)}\big[c(I, C) \mid C\big] = \sum_{d=1}^{14} q(D = d \mid \bar{I} = 1, O, S, E) \cdot \bar{c}_d(C). \tag{18}$$

Average over C For d fixed, the distribution of $\bar{c}_d(C)$ can be very complex due to the distribution over C (i.e. a distribution over piecewise linear functions). We can estimate the expectation over the contagiousness profile C using Monte-Carlo estimation:

$$\mathbb{E}_{q(C|O,S,E)}\big[\bar{c}_d(C)\big] \approx \frac{1}{m} \sum_{j=1}^{m} \bar{c}_d(C_j) \qquad C_j \overset{\text{iid}}{\sim} q(C \mid O, S, E). \tag{19}$$

Overall, this means that the average contagiousness (to be quantized for the risk level) can be estimated with

$$\mathbb{E}_{q(I,C|O,S,E)}[c(I, C)] \approx \frac{1}{m} \sum_{j=1}^{m} \sum_{d=1}^{14} q(D = d \mid \bar{I} = 1, O, S, E) \cdot \bar{c}_d(C_j). \tag{20}$$

3.4.2 Risk level updates

In the previous section, we saw how to compute today's risk level. However since we have access to the full contagiousness profile, it is possible to compute the contagiousness (and therefore the risk level) at any point in the past by shifting Equation 20. The risk level for δ days in the past is

$$\text{Risk level}(-\delta) = \text{quantize}\left[\frac{1}{m} \sum_{j=1}^{n} \sum_{d=\delta+1}^{14} q(D = d \mid \bar{I} = 1, O, S, E) \cdot \bar{c}_{d-\delta}(C_j)\right]$$
$$C_j \overset{\text{iid}}{\sim} q(C \mid O, S, E). \tag{21}$$

We can use this risk level to send an update message to an encounter in the past, based on an updated estimate of the contagiousness, to refine the estimation of the risk for the other user.

4 Training

The app on each phone will get observations (S, E, O) every day; recall that E and O contains all the observations of encounters and symptoms / test results for the past 14 days. Based on these observations, we want to learn both the parameters θ of the generative model $p_\theta(I, C, O \mid S, E)$ and the parameters ϕ of the inference model $q_\phi(I, C \mid O, S, E)$.

4.1 Amortized variational inference

Similar to Variational Autoencoders (VAEs), we can use amortized variational inference to jointly train the generative and inference models. We can write the Evidence Lower-Bound (ELBO) for the conditional log-likelihood $\log p(O \mid S, E)$ (recall that this is a conditional model):

$$\begin{aligned}\log p_\theta(O \mid S, E) \geq \mathbb{E}_{q_\phi(I,C|O,S,E)}\big[\log p_\theta(O \mid I, C, S, E)\big] \\ - \mathrm{KL}\left(q_\phi(I, C \mid O, S, E) \,\|\, p_\theta(C \mid S) p_\theta(I \mid S, E)\right)\end{aligned} \tag{22}$$

The ELBO contains two terms, the *expected log likelihood* and the *KL divergence*, both of which we will detail further in the following sections.

4.1.1 Expected log-likelihood

The reason why we chose a Categorical distribution to model $q(I \mid O, S, E)$ in Section 3.3 was to simplify the computation of the expected log-likelihood term in the ELBO; when summing over I, we only have a number of terms linear in n, as opposed to exponential in n with a vector of Bernoulli variables. Since the probability of symptoms & test results was defined in terms of $\bar{I}$ and D in Section 2.3.2, we can write the expected log-likelihood as

$$\begin{aligned}&\mathbb{E}_{q_\phi(I,C|O,S,E)}\big[\log p_\theta(O \mid I, C, S, E)\big] \\ &\quad = q_\phi(\bar{I} = 0 \mid O, S, E)\, \mathbb{E}_{q_\phi(C|O,S,E)}\big[\log p_\theta(O \mid \bar{I} = 0, C, S)\big] \\ &\quad + \sum_{d=1}^{14} q_\phi(D = d \mid \bar{I} = 1, O, S, E)\, \mathbb{E}_{q_\phi(C|O,S,E)}\big[\log p_\theta(O \mid D = d, \bar{I} = 1, C, S)\big],\end{aligned} \tag{23}$$

where the distributions $q(\bar{I} = 0 \mid O, S, E)$ and $q(D = d \mid \bar{I} = 1, O, S, E)$ are defined in Equations 14 and 15 respectively.

Reparametrization trick We can use the reparametrization trick to estimate, with Monte-Carlo, the remaining expectations over $q(C \mid O, S, E)$ in Equation 23.

- **Log-Normal distribution**: We can apply the reparametrization trick to compute the expectation of a function f (with parameters θ) over a log-Normal distribution $q = \log \mathcal{N}(\mu, \sigma^2)$ with

$$\mathbb{E}_{q(z)}\big[f(z\,;\,\theta)\big] = \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,1)}\big[f(\exp(\mu+\sigma\varepsilon)\,;\,\theta)\big]. \tag{24}$$

- **Uniform distribution**: We can apply the reparametrization trick to compute the expectation of a function f (with parameters θ) over a Uniform distribution $q = \mathcal{U}(a, b)$ with

$$\mathbb{E}_{q(z)}\big[f(z\,;\,\theta)\big] = \mathbb{E}_{\varepsilon\sim\mathcal{U}(0,1)}\big[f(a+(b-a)\varepsilon\,;\,\theta)\big]. \tag{25}$$

4.1.2 KL divergence

Given the structure of the inference model (see Figure 1b), the KL divergence term can be further decomposed into two separate terms for I and C:

$$\begin{aligned}&\mathrm{KL}\left(q_\phi(C \mid O,S,E)q_\phi(I \mid O,S,E)\,\|\,p_\theta(C \mid S)p_\theta(I \mid S,E)\right) \\ &\quad= \mathrm{KL}\left(q_\phi(C \mid O,S,E)\,\|\,p_\theta(C \mid S)\right) + \mathrm{KL}\left(q_\phi(I \mid O,S,E)\,\|\,p_\theta(I \mid S,E)\right)\end{aligned} \tag{26}$$

KL divergence in C Recall that the random variable C represents the contagiousness profile, and consists of 5 continuous values; see Section 2.1.2 for details. Since these 5 random variables are conditionally independent, we can even further decompose the KL-divergence in C into 5 separate terms, one for each univariate variable. Since the distributions are either (truncated) Normal or Uniform distribution, we can compute each term in closed form.

- **Log-Normal distributions**: The KL-divergence between two (univariate) log-Normal distributions $q = \log \mathcal{N}(\mu_1, \sigma_1^2)$ and $p = \log \mathcal{N}(\mu_2, \sigma_2^2)$ is given by

$$\mathrm{KL}(q\,\|\,p) = \log\frac{\sigma_2}{\sigma_1} + \frac{(\mu_1-\mu_2)^2+\sigma_1^2-\sigma_2^2}{2\sigma_2^2}. \tag{27}$$

- **Uniform distributions**: The KL-divergence between two Uniform distributions $q = \mathcal{U}(a, b)$ and $p = \mathcal{U}(c, d)$ is finite only if $c \le a < b \le d$ and is given by

$$\mathrm{KL}(q\,\|\,p) = \log\frac{d-c}{b-a}. \tag{28}$$

KL divergence in I Since the variable I is a discrete random variable, the KL divergence can be written explicitly as a weighted sum

$$\begin{aligned}\mathrm{KL}\left(q_\phi(I \mid O,S,E)\,\|\,p_\theta(I \mid S,E)\right) &= q_\phi(I=0 \mid O,S,E)\log\frac{q_\phi(I=0 \mid O,S,E)}{p_\theta(I=\mathbf{0} \mid S,E)} \\ &+ \sum_{t=1}^{n+14} q_\phi(I=t \mid O,S,E)\log\frac{q_\phi(I=t \mid O,S,E)}{p_\theta(I_t=1 \mid S,E)} \\ &+ \big(1-q_\phi(I=0 \mid O,S,E)\big)\cdot\left[\log\sum_{t=1}^{n+14} p_\theta(I_t=1 \mid S,E) - \log\big(1-p_\theta(I=\mathbf{0} \mid S,E)\big)\right],\end{aligned} \tag{29}$$

where $p(I = \mathbf{0} \mid S, E)$, given in Equation 7, is the probability of not being infected. Recall that in the generative model, I is a vector of binary variables, whereas in the inference model, I is a categorical random variable, hence the distinction between "$I_t = 1$" in p and "$I = t$" in q. The last term in Equation 29 corresponds to the normalization over p, and assumes a uniform prior of infection over the different encounters (measured and unmeasured).

4.1.3 Training objective

Since we want to maximize the conditional log-likelihood $\log p(O \mid S, E)$, we can maximize the ELBO with respect to jointly the parameters of the generative model θ and those of the inference model ϕ

$$\min_{\theta,\phi} \mathcal{L}(\theta, \phi; \mathcal{D}), \tag{30}$$

where $\mathcal{D} = \{O_i, S_i, E_i\}_{i=1}^N$ is a dataset of observations (from either the simulator, or from the real world), and the objective is defined as

$$\mathcal{L}(\theta, \phi; \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \Bigg[\underbrace{\mathrm{KL}\left(q_\phi(I, C \mid O_i, S_i, E_i) \,\|\, p_\theta(C \mid S_i) p_\theta(I \mid S_i, E_i)\right)}_{\text{Section 4.1.2}} - \underbrace{\mathbb{E}_{q_\phi(I,C \mid O_i, S_i, E_i)}\left[\log p_\theta(O_i \mid I, C, S_i, E_i)\right]}_{\text{Section 4.1.1}} \Bigg]. \tag{31}$$

4.2 Wake-sleep algorithm

The Wake-sleep algorithm can be viewed as an approximate implementation of the Expectation Maximization algorithm (EM). It consists in two phases: the *wake phase*, and the *sleep phase*.

- **Wake phase**: From the observations O, S, and E, sample the latent variables I and C from the current inference model. Then use this fully observed data (I, C, O, S, E) to maximize the likelihood of the generative model $p(I, C, O \mid S, E)$.
- **Sleep phase**: This time only based on the observations S and E, sample "fantasy" data from the current generative model $p(I, C, O \mid S, E)$. Then use this fully observed data (I, C, O, S, E) to maximize the likelihood of the inference model $q(I, C \mid O, S, E)$.

The wake and sleep phases are applied iteratively. In the following sections, we will detail the implementation of both phases, based on a dataset of observations $\mathcal{D} = \{O_i, S_i, E_i\}_{i=1}^N$.

4.2.1 Wake phase

The wake phase operates in two steps: first the data is completed using the inference model, and then the parameters of the generative model are optimized.

1. For every datapoint (O_i, S_i, E_i), we sample m possible completions from the current inference model $q_{\phi^{(t)}}(I, C \mid O, S, E)$:

$$I_{i,j}, C_{i,j} \overset{\text{iid.}}{\sim} q_{\phi^{(t)}}(I, C \mid O_i, S_i, E_i). \tag{32}$$

2. Using the fully observed data $(I_{i,j}, C_{i,j}, O_i, S_i, E_i)$, we can maximize the log-likelihood of the completed data wrt. the different parameters of the generative model:

$$\theta^{(t+1)} = \arg\max_{\theta} \frac{1}{Nm} \sum_{i=1}^{N} \sum_{j=1}^{m} \log p_{\theta}(I_{i,j}, C_{i,j}, O_i \mid S_i, E_i). \tag{33}$$

4.2.2 Sleep phase

The sleep phase also operates in two steps: first "fantasy" data is generated from the generative model, and then the parameters of the inference model are optimized.

1. Since we have a generative model conditioned on S and E, for every datapoint (S_i, E_i), we sample m possible datapoints from the current generative model $p_{\theta^{(t+1)}}(I, C, O \mid S, E)$:

$$I_{i,j}, C_{i,j}, O_{i,j} \overset{\text{iid.}}{\sim} p_{\theta^{(t+1)}}(I, C, O \mid S_i, E_i). \tag{34}$$

2. Using the fully observed data $(I_{i,j}, C_{i,j}, O_{i,j}, S_i, E_i)$, we can maximize the log-likelihood of the latent variables I and C wrt. the parameters of the inference model:

$$\phi^{(t+1)} = \arg\max_{\phi} \frac{1}{Nm} \sum_{i=1}^{N} \sum_{j=1}^{m} \log q_{\phi}(I_{i,j}, C_{i,j} \mid O_{i,j}, S_i, E_i). \tag{35}$$

5 Mobility model

In order to build a fully functional simulator, we need to add a *mobility model* on top of the generative model from Figure 1a and Section 2. This mobility model, $p(E \mid S)$, is responsible for generating encounters in E. To enable efficient sampling for a large population (e.g. a whole country like Canada, with 30M people), we propose a model which operates at two different levels.

5.1 Mobility model within a central zone

At the level of a single zone, we can run an *agent-based* simulation over the whole population within that zone. Let's say that we place ourselves in a zone Z, with a population of n users. The goal here is to create a contact graph between these n users that can later be used to sample the encounters inside Z.

We model this by independently sampling the edges of the graph according to the following probability distribution over contacts Y_{uv}

$$p(Y_{uv} \mid N_{uv}, R_u, R_v, S_u, S_v, H_u, H_v, Z), \tag{36}$$

conditioned on information from both users u and v. The notations are defined in Table 3. To define this probability distribution though, we would need access to joint observations from both u and v, which we do not have for privacy reasons; we only have access to the risk level of the other user, as well as the number of repeated encounters, through the messages received and the output of the clustering algorithm. Moreover, the location data Z and medical data H and S are stored in different records; there is no joint observations between location and medical data either, to avoid any identification.

Variable	Type	Description
Y_{uv}	Binary	The **contact event** between users u and v. $Y_{uv} = 1$ means there has been a contact between u and v; this is symmetric ($Y_{uv} = Y_{vu}$).
N_{uv}	Categorical	The **number of repeated encounters** between u and v, as estimated by the clustering algorithm.
S_u	–	**Static information** about user u (e.g. demographic, pre-existing medical conditions). Same as variable S from Section 2.
R_u	Categorical	The current **risk level** of user u. This is the quantized version of the contagiousness returned by the inference model (see Section 3.4)
H_u	Integer matrix	The **historical data** of user u for the past 14 days. This is a matrix containing the number of encounters with specific risk levels and number of repetitions.

Table 3: Notations for the mobility model within a central zone. Some of these notations are also used in the mobility model with neighbor zones (see Section 5.2).

As such, we need to estimate the probability distribution from Equation 36 with quantities we can eventually learn from data. We propose the following decomposition to mitigate bias in our estimation:

$$p(Y_{uv} \mid R_u, R_v, S_u, S_v, H_u, H_v, Z) \approx \tag{37}$$
$$\Big[\underbrace{q(Y_{uv} \mid N_{uv}, R_u, S_u, H_u, R_v) \cdot q(Y_{uv} \mid N_{uv}, R_v, S_v, H_v, R_u)}_{\text{Section 5.1.2}} \cdot \underbrace{q(Y_{uv} \mid Z, R_u, R_v)}_{\text{Section 5.1.3}}\Big]^{1/3}.$$

5.1.1 Historical data

A strong predictor of the mobility pattern comes from historical data of encounters the user had in the past 14 days. This data is derived from the observations of encounters E stored on the phone of the user. What we are interested in is aggregated statistics about the encounters of user u, broken down by

1. *Risk level* r, for all the possible risk levels (currently 16 values). The risk levels are contained in the messages received by u

2. *Number of repetitions* m, categorized by frequency of encounters in the past 14 days. For example, we can use 5 categories: 1 encounter (non-repeated encounter), $[2, 4)$ encounters, $[4, 8)$ encounters, $[8, 16)$ encounters, and 16+ encounters. The number of repetitions can be approximated using the clustering algorithm of the messages received.

We can therefore create a matrix histogram H_u, whose entries $H_u(r, m)$ contains the number of encounters of risk r, m with repeated encounters, in the past 14 days. In the following sections, we will assume that the matrix H_u has size 16×5.

To build this histogram, we need to call the clustering algorithm. However the clustering algorithm might cluster together encounters with varying risk levels (e.g. I met someone twice, once she was risk level 2, and the second time she was risk level 4). Should these clusters be broken down further by risk level?

State of a zone We might have some users with very little data available (e.g. a new user), for which H_u might not be informative enough. We can aggregate the historical data of all the users in the zone Z to estimate its state, i.e. what is the typical mobility pattern for users in Z.

$$H(Z) = \sum_{u \in Z} H_u. \tag{38}$$

5.1.2 Probability of contact given user information

Given only data for user u, we want to model the probability that u has a contact with another user v with specific risk levels and number of repeated encounters. In other words, we want to model

$$q(Y_{uv} \mid N_{uv}, R_u, S_u, H_u, R_v). \tag{39}$$

Alternatively, we can model the probability of contact for all possible combination of risk levels and number of repetitions at once, and then select the entries we are interested in. Concretely, we want to model $\mathbf{Y}$ as a matrix of binary values, where $\mathbf{Y}$ has the same size as the historical data H_u (16×5, see Section 5.1.1). We can use a neural network with inputs R_u, S_u and H_u (*normalized* histogram), with parameters θ:

$$q(\mathbf{Y} = 1 \mid R_u, S_u, H_u) = \sigma\big(f(R_u, S_u, H_u\,;\,\theta)\big). \tag{40}$$

Since we have matrix inputs and outputs, both having the same size, we can use a convolutional neural network as our function f. The risk level R_u and additional static information S_u can be added as separate channels to the input H_u. Figure 7 shows this convolutional network, with its inputs and outputs.

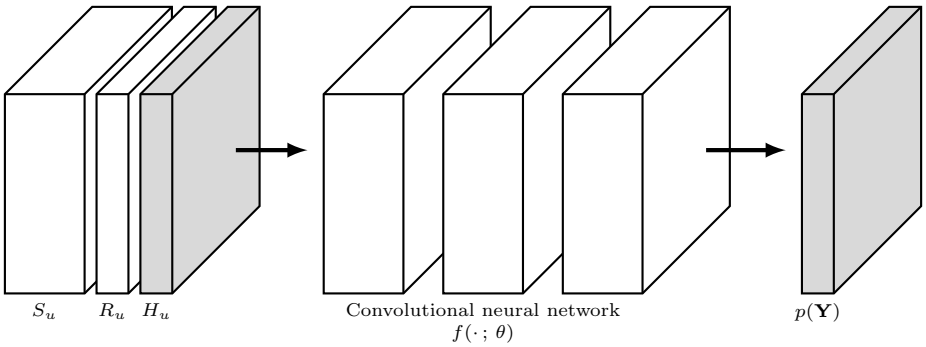


Figure 7: Schematic view of the convolutional neural network architecture used for the probability distribution in Equation 40.

To obtain the probability distribution of u having a contact with v, given the risk level r and number of repetitions m of v (Equation 39), we can select the corresponding entry in $\mathbf{Y}$:

$$q(Y_{uv} = 1 \mid N_{uv} = m, R_u, S_u, H_u, R_v = r) = q(\mathbf{Y}[r, m] = 1 \mid R_u, S_u, H_u). \quad (41)$$

Training We can train this neural network using supervised learning (maximum likelihood) on observations available on the phone of each user u. More precisely, we can gather a dataset $\mathcal{D} = \{S_u, R_u, H_u, \mathbf{Y}_u\}_u$ for all users (from not only zone Z, but from all the zones).

Here, we have to distinguish more precisely the parts of historical data is treated as input and output; both of these can be derived from the observations in E (i.e. the messages received on the phone, and the subsequent result of the clustering algorithm)

- The input H_u of the neural network contains the historical data for the past 14 days, as described in Section 5.1.1, *excluding today's encounters.*
- The target $\mathbf{Y}_u$ of the neural network is a matrix of *binary outcomes of today's encounters.* In other words, $\mathbf{Y}_u[r, m] = 1$ if user u has been in contact today with at least one user with risk level r and repetition m.

The loss function used for supervised learning is simply the binary cross entropy between the output of the neural network and the target $\mathbf{Y}_u$

$$\mathcal{L}(\theta\,;\mathcal{D}) = \sum_u \text{cross-entropy}(\mathbf{Y}_u, \hat{\mathbf{Y}}_u) \quad (42)$$
$$\text{where } \hat{\mathbf{Y}}_u = \sigma\big(f(R_u, S_u, H_u\,;\theta)\big).$$

5.1.3 Probability of contact given the zone

TODO: Complete this section

5.2 Mobility model with a neighbor zone

To model mobility between different zones, we also have to model how users behave outside of their central zone. Contrary to Section 5.1, where we were running a full agent-based model at the level of the central zone (with a contact graph, and users u and v being real agents from the zone), here we want to model an encounter between a user u and *a generic (fictitious) agent* v with a specific risk level R_v and number of repetition N_{uv} in another zone

$$p(Y_{uv} \mid N_{uv}, R_u, R_v, S_u, S_v, H_u, L_u, Z_u, Z_v, Z_{\text{contact}}). \tag{43}$$

Additional notations are defined in Table 4. For reasons similar to the ones mentioned in Section 5.1, namely the separation of the records for privacy reasons, we will estimate this probability distribution using a decomposition similar to Equation 37

$$\begin{aligned} p(Y_{uv} \mid N_{uv}, R_u, R_v, S_u, S_v, H_u, L_u, Z_u, Z_v, Z_{\text{contact}}) \approx \\ \Big[\underbrace{q(Y_{uv} \mid N_{uv}, R_u, S_u, H_u, R_v)}_{\text{Section 5.1.2}} \cdot \underbrace{q(Y_{uv} \mid Z_u, Z_{\text{contact}})}_{\text{Section 5.2.1}} \\ \cdot \underbrace{q(Y_{uv} \mid N_{uv}, R_v, Z_v)}_{\text{Section 5.2.2}} \cdot \underbrace{q(Y_{uv} \mid Z_v, Z_{\text{contact}})}_{\text{Section 5.2.1}} \Big]^{1/4}. \end{aligned} \tag{44}$$

The first term of this decomposition has already been described in Section 5.1.2 in the case of mobility modeling within a central zone.

Variable	Type	Description
Y_{uv}	Binary	The **contact event** between user u and a *fictitious user* v, defined by its risk level and number of repeated encounters with u.
Z_u	–	The **central zone** of user u. This typically corresponds to the zone where u lives.
Z_{contact}	–	The **zone where the contact** between u and v happens. This can be different from the central zones of u and v.

Table 4: Additional notations used for the mobility model with a neighbor zone.

5.2.1 Probability of transition between zones

To model the probability that user u will move from its central zone Z_u to Z_{contact}, we can use aggregated location data contained in the records for zone Z_u to estimate the transition probability distribution

$$q(Y_{uv} \mid Z_u, Z_{\text{contact}}). \tag{45}$$

For now, Z_u and Z_{contact} are taking values in a list of zone IDs. Eventually, we can have a zone embedding representing each zone, based on information about the zone (e.g. demographics, average risk, location).

5.2.2 Probability of fictitious agent

Since we are not precisely monitoring the other user v whom u has an encounter with in another zone (i.e. there is no explicit contact graph involving u and v, unlike in the central zone in Section 5.1), we have to define a probability distribution that this fictitious user v exists, as defined by its risk level R_v and the number of repeated encounters N_{uv} with u

$$q(Y_{uv} \mid N_{uv}, R_v, Z_v). \tag{46}$$

For this, we can use aggregated statistics about the risk levels in zone Z_v, as described in Section 5.1.1 and Equation 38. Similar to how we used a convolutional neural network in Section 5.1.2, we can define the probability of $\mathbf{Y}$ for all risk levels and number of repetitions

$$q(\mathbf{Y} \mid Z_v) = \sigma\big(g(H(Z_v)\,;\,\theta)\big), \tag{47}$$

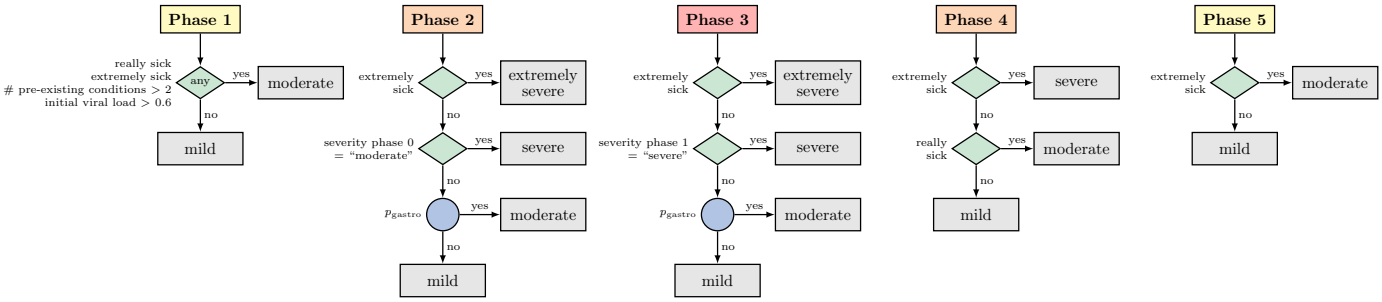
where g is a CNN with parameters θ, and $H(Z_v)$ is the aggregated histograms (normalized) for all users from zone Z_v. We can also train this neural network using supervised learning.

Appendix

Version of the code All the links to existing code in the document are based on the following versions of the projects:

- `covid_p2p_simulation`: 62c7e41
- `covid_p2p_risk_prediction`: 8cf2e28

A Symptoms generation in the simulator



code code code code code

Figure A.1: Logic of the severity of the symptoms in the simulator, based on the phases defined in Figure 4. Diamond green nodes are (deterministic) decision nodes, and round blue nodes are stochastic nodes ("yes" with probability p).