Estimation of Infection Risk for COVID-19

Editor:

Abstract

In the context of automated contact tracing, we need a way to estimate infection risk of an individual given their history of contacts with other individuals who may or may not be infected and for which some estimation of risk is also available. Once enough data is collected this can be done by machine learning methods but we also need a starting heuristic based on epidemiological considerations.

1. Infection Risk Estimation from Contacts

Consider an individual whose phone is collecting data about close encounters with other individuals, where their phones can communicate information peer-to-peer in order to help each other update their infection risk. We define risk here simply as being infected. In practice we want to predict the probabilities for a variety of possible event which are not exactly the same as being infected or not, because infection tests are imperfect and can have as much as 30% or 40% of the negatives be false negatives. For now however let us consider binary events B_{ti} which are observed at time t for individual i (e.g. $B_{ti} = 1$ means the test would be positive and $B_{ti} = 0$ means would be negative). The risk level is then defined as $r_{ti} = P_\theta(B_{ti} = 1|h_{ti})$ which is an estimator of the true probability with parameters θ, and where h_{ti} is an information set available just before time t.

1.1 Contacts

When two individuals i and j get close to each other in some time interval (t_1, t_2) we will measure some quantities which could be predictive of the probability that i infects j given that j is infected and vice-versa. Multiplying by the probability that they are infected would give us an estimator of a contagion event. Let e_k be summarizing the information about the contact event k involving individuals i and j, then $h_{ti} = \{e_k \text{ involving } i,\ t - C_{max} < t' < t\}$ the set of such events which involve i and happen between $t - C_{\max}$ and t, where C_{max} is the maximum period of being contagious after having been infected. We will also include medical events in

2. Non-ML Baseline

Before a tracing app is deployed with don't have much detailed data of the kind necessary to directly estimate the risk level by ML but we could use independence assumptions and aggregate statistics in order to construct a baseline.

Let p_k be the probability that i could infect j if i is contagious and vice-versa (we will assume symmetry here). Then, assuming independent events,

$$r_{t+1,i} = 1 - (1 - r_{t,i})(1 - p_k r_{t,j}). \tag{1}$$

which arises simply from the definition of these probabilities and assuming that the events e_k are independent of each other. Lenka: This equations is equivalent to our eq. (11) after mapping $r_{t,i} = P_I^i(t), p_k = \lambda_{i,j}(t), P_R^i(t) = 0$ and taking times steps such that there is only one contact at every time step.

2.1 Contact events

The p_k (conditional transmission probability) is a quantity that can either be estimated based on statistics collected by epidemiologists, or better estimated by a machine learning model (see below). The inputs would be things like the duration of the contact, how long the two persons spent at different distances (as a histogram). One may want to take into account that transmission could happen even if the persons were not at the same place at the same time but a few minutes apart, since the virus can linger in the air for a few minutes in water droplets.

2.2 Medical events

In addition to contact events, other e_k's could be associated with specific novel medical information (such as test results), and one would thus estimate the conditional probability of being infected a posteriori (given these observations and the past probability of being infected). This probably requires ML to do well. A simple ML-free version could simply consider the infection test result, setting $r_{t+1,i}$ according to the aggregate statistics of true positives relative to the number of positives for that test.

2.3 Dependent Events

The easy case is if the e_k events are independent, yielding a solution like in Eq. 1. How do we handle the case of dependent events. These occur when the same person is encountered multiple times, for example. What we suggest to handle this, short of analytic examples, is to run Monte-Carlo simulations where we keep track of and sample the random variables associated with the (binary) infection status of each person and the (binary) transmission event when two persons have a contact. We can use the same conditional probabilities for transmission and for medical events but now simulate that process over a network of individuals (using epidemiological assumptions for the connectivity of that network, about the distribution in time and network structure of contacts).

Not being able to run a full MC simulation across everyone for privacy reasons, we may want to make some approximations to the full state-space (of beliefs over the joint distribution of everyone's infection status).

An obvious ingredient in such approximations is that we should distinguish the frequent contacts (e.g. people living in the same household) from the others. Maybe a way to deal with that is to think of the risk update as something that pushes the min of the two risks towards the max, in proportion to the difference times a scalar which corresponds to the probability of contagion:

$$r_{t+1,i} = r_{t,i} + (r_{t,j} - r_{t,i})p_k \tag{2}$$

looks like a typo to me: form eq (1), this should be this; did I miss anything?

$$r_{t+1,i} = 1 - (1 - r_{t,i})(1 - p_k r_{t,j}) = \tag{3}$$
$$r_{t,i} + (1 - r_{t,i})p_k r_{t,j} = \tag{4}$$
$$r_{t,i} + (r_{t,j} - r_{t,i} r_{t,j})p_k \tag{5}$$

assuming wlog that $r_{t,i} < r_{t,j}$. Comparing with eqn 1 we can see it is actually the same. The only difference is that we only want to update the risk of the person with the smallest risk.

Also, instead of thinking of what each person's prediction should be given the information from its contacts and transmission risks at each contact event, we can think of the global computation going on as persons exchange information about their risk status. This computation could be made to look like a loopy belief propagation, which estimates the marginal probabilities of each local random variable given the marginal probabilities in its neighbors.

In order to test these ideas, we need to do a Monte-Carlo simulation where we run the full sampling of a whole population with some meeting probability distribution and some contagion probabilities and see if these estimators work well or not.

The MC simulation could be structured as follows. We associate to each person a position $X_{t,i}$ and we create a reasonable random walk for people, maybe using a 'home address' on a map (with 1-5 people in each household). People either stay at home with some probability or do a spatial random walk outside, then come back home. We infect the population initially randomly by picking a subset of it (in reasonable proportions based on the current stage of the epidemics, like 0.1% or 0.5%) We have at each time step some set of 'meetings' when two persons are within some radius of each other and that provides the data for our risk estimators. We unfold this in time for a few weeks. We apply some estimation method (ML or not) for the risk at each time step (putting some lags for propagation) and we compare the methods together in terms of how well they estimate the risks.

3. ML Candidates

Once we get access to actual individual level data about detection tests, we can train a risk prediction level to predict the results of those tests based on the history of contacts and

medical events. Note that the only necessary inputs are not about location of people but about the risk levels associated with each contact, as well as about the test results.

The ML prediction could be structured as above, i.e., we only need to parameterize the contagion probability estimator. However, it would be tuned end-to-end so that we correctly predict the test results probabilities (maximum likelihood).

Note that the input to the predictor for one app (one person) depends on the output of the predictor for other apps (people with whom the person was in contact), so there are subtle interactions which need to be carefully thought about between what different predictors are doing. The good news is that we are considering centralized learning so all the apps will be running the same algorithm.

A good framework for thinking about this is probably that of loopy belief propagation in undirected graphical models (factor graphs). What this also means is that when some 'news' hits the network of contacts (someone got tested, someone has symptoms) the information needs to propagate iteratively to approximately converge globally. Each node in the graph will update its belief (about being infected or not) by exchanging messages with its neighbors in the graph (which are its active contacts). My active contacts are those which might have influenced me (e.g. contacts which happened in the last 14 days).

The core ingredient of the factor graph is the potential function associated with an edge. Here an edge corresponds to the interaction e_k that took place in the past between person i and person j, in come circumstances stored in a vector associated with e_k. We want to parameterize that potential function and learn it. What are the good ways of doing that which can scale to a graph with tens of millions of nodes? (or more). Simple local methods are probably preferred.

4. Algorithm

5. Appendix: Notes from Brainstorming Sessions

Pseudocode: Tegan's notes

```
# This is a minimal version which only uses the features:
#    - tested or not
#    - severity of symptoms (if any)
#    - time since infection

#memoryless version
transmission_prob = 0.25

#memory version
infectiousness_curve =  [0.05, 0.1, 0.2, 0.3, 0.25, 0.2, 0.15, 0.1, 0.05]
if day_infected > len(infectiousness_curve)
    transmission_prob = infectiousness_curve[-1]
else
```

```
    transmission_prob = infectiousness_curve[day_infected]


if self.test_result = ‘positive’:
    self.risk = 1
elif self.symptoms = ‘severe’:
    self.risk = 0.8
elif self.symptoms = ‘moderate’:
    self.risk = 0.5
elif self.symptoms = ‘mild’:
    self.risk = 0.25

#Yoshua’s version
if contact:
    if self.risk < other.risk:
        self.risk += (other.risk - self.risk)*transmission_prob

#Lenka’s version
if contact:
    self.risk += other.risk*transmission_prob



Position à toutes les 5 minutes, plus incertitude sur la posn



Other notes

#Len
Inputs: history of interactions with approximate ids
of the people contacted, with time stamps, risks of the other people,
eventually coarse geographical location (1/1000 people precision)
Output: updated risk level(s)

Handcrafted
 - covid-specific symptoms or not
 - test results or not
 - had a contact or not (and that person’s level of risk)

 - if you have contact with someone who has a + test result,
 risk level for all their contacts goes up by an amount
 proportional to difference in risk level * p(transmission)
```

References