Online Appendix — COVID-19: Prediction, Prevalence, and the Operations of Vaccine Allocation

Mohammed Amine Bennouna
Operations Research Center, MIT, Cambridge, MA 02139, amineben@mit.edu

Joshua Joseph
Quest, MIT, Cambridge, MA 02139, jmjoseph@mit.edu

David Nze-Ndong
Sloan School of Management, MIT, Cambridge, MA 02139, davidzedong@gmail.com

Georgia Perakis
Sloan School of Management, MIT, Cambridge, MA 02139, georgiap@mit.edu

Divya Singhvi
Stern School of Business, New York University, New York, NY 10012, divya.singhvi@stern.nyu.edu

Omar Skali Lami
Operations Research Center, MIT, Cambridge, MA 02139, oskali@mit.edu

Yiannis Spantidakis
Operations Research Center, MIT, Cambridge, MA 02139, yspant@mit.edu

Leann Thayaparan
Operations Research Center, MIT, Cambridge, MA 02139, lpgt@mit.edu

Asterios Tsiourvas
Operations Research Center, MIT, Cambridge, MA 02139, atsiour@mit.edu

---

Appendix A: Minimum Representation Learning Model

In this section we describe further the Minimal Representation Learning algorithm (`MRL`). We refer the reader to Bennouna et al. (2021) for a detailed description of the general approach and theoretical analysis.

`MRL` uses the observed data from the dynamic system to partition the feature space into states of a finite deterministic MDP $(\mathcal{S}, \mathcal{A}, f, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and $r : \mathcal{S} \to \mathbb{R}$ are the transition and outcome functions. Each feature $x \in \mathbb{R}^N$ is mapped to a unique MDP's state $\phi(x) = s \in \mathcal{S}$ with a mapping $\phi : \mathbb{R}^N \longrightarrow \mathcal{S}$. Figure 1 illustrates this process. The constructed MDP constitutes a concise reduced representation of the system that approximates accurately the system's values, and therefore models COVID-19's evolution in our case.

Once the model is constructed, we can predict values of the dynamic system at features x and taking a sequence of actions $a_1, \ldots, a_h \in \mathcal{A}$ (e.g., restrictive mobility measures), by mapping the feature vector x to its

corresponding state $s := \phi(x)$ in the MDP representation and then extracting the value of the representation MDP starting at s and taking actions $a_1, \ldots, a_H$, i.e.,

$$\hat{V}(x) = r(f(s)) + \sum_{t=1}^{h} r(f(s, a_1 \ldots a_t)), \tag{1}$$

where $\hat{V}(x)$ is the predicted value and $f(s, a_1 \ldots a_t) = f(\ldots f(f(s, a_1), a_2) \ldots, a_t)$.

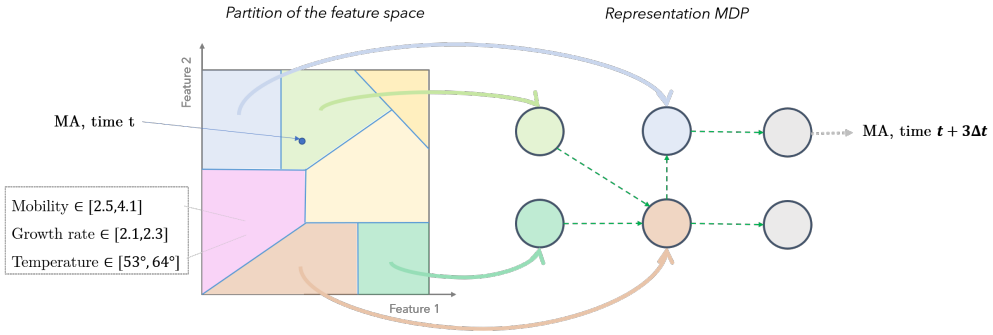

Figure 1 **Illustration of the representation MDP construction. The feature space is partitioned into a finite number of regions (left). A representation MDP is then constructed and each region is mapped (full arrows) to a state of the MDP (right). To make a prediction, a region at a given date is mapped with its features to a region of the feature space, then mapped to the corresponding state in the representation MDP.**

To learn this concise MDP representation using transition data, the `MRL` algorithm iteratively splits the feature space to approximate an MDP representation that is consistent with the transition data. Such a representation is called *coherent* (see Definition 4.1 in Bennouna et al. 2021). More precisely, a mapping of the feature space $\hat{\phi}$ into states of an MDP is defined to be coherent if any two data points (feature vectors) are mapped to the same state and share the same outcome, i.e.,

$$\hat{\phi}(X_{i_1,t_1}) = \hat{\phi}(X_{i_2,h_2}) \implies R_{i_1,t_1} = R_{i_2,t_2}, \quad \forall i_1, i_2, \forall t_1, t_2.$$

and every two data points mapped to the same state and having observed the same action transition to data points mapped to the same state, i.e.,

$$\hat{\phi}(X_{i_1,t_1}) = \hat{\phi}(X_{i_2,t_2}) \text{ and } A_{i_1,t_1} = A_{i_2,t_2} \implies \hat{\phi}(X_{i_1,t_1+1}) = \hat{\phi}(X_{i_2,t_2+1}), \quad \forall i_1, i_2, \forall t_1, t_2.$$

Bennouna et al. (2021) show that any coherent representation of the system converges to the most concise (with smallest number of states) representation of the system. Hence, the `MRL` method learns the most concise MDP representation of the COVID-19 evolution. The `MRL` approach also provides provable guarantees as we summarize in Theorem 1 below.

THEOREM 1 **(see Bennouna et al. (2021))**. *Let N be the number of sample paths and T be the length of each path in the training data. Let V be the true value function of the system and $\hat{V}$ be the value estimated*

as in (1) *for a prediction horizon* $h \in \mathbb{N}$ *with a coherent MDP representation. For all* $\delta > 0$*, with probability* $1-\delta$*,*

$$|\hat{V} - V| = O\left(\sqrt{\frac{h}{NT}\left[\log(NT) + \log\left(\frac{1}{\delta}\right)\right]}\right).$$

We note that the same bound holds when the prediction is of the form $e^{\hat{V}}$, which is the case in COVID-19. That is, asymptotically $|e^{\hat{V}} - e^{V}| = O(e^{|\hat{V}-V|} - 1) = O(\hat{V} - V)$.

Model Formulation In this section, we detail how we model COVID-19 evolution as a dynamic system. The prediction target $Y_{i,t}$ for a given region i, at day t can be either the *cumulative number of cases* or the *cumulative number of deaths*.

Introducing a time discretization. The day-to-day growth rate, cases and deaths are volatile due to fluctuations of testing numbers and seasonality among others. As a result, aiming to make accurate daily predictions is a complex problem. Furthermore, such a precise number is not necessary for the purpose of policy making. Instead of learning a day-to-day outcome, we introduce a number of days d onto which we *aggregate* the observations. Hence, for some arbitrary starting time t_0, we consider the aggregated outcomes on the set of times $t \in \{t_0, t_0 + d, t_0 + 2d, \ldots\}$ defined as

$$Y^{(d)}_{i,h} = \frac{1}{d}\sum_{\tau=0}^{d} Y_{i,t_0+ht-\tau}, \quad \forall h \leq T/d$$

where d is chosen as a trade-off between *learnability* and *precision*. Moreover, we observe some seasonal behavior on a short-term in the target time series. Averaging the curve allows us to recover the trend in a more stable and easier way that translates into an MDP.

From growth rate to outcomes. We define the MDP costs R as the logarithm of the aggregated growth rate:

$$R_{i,h} = \log\left(Y^{(d)}_{i,h+1} / Y^{(d)}_{i,h}\right).$$

Suppose the goal is to predict the outcome in region i for a horizon $h_0 + H$, where the available data stops at time h_0. We use the constructed finite MDP representation by `MRL` to predict sequences of costs of the MDP — i.e., log aggregated growth rates — $\{\hat{R}_{i,h_0+h},\, h \in [1, H]\}$, deduce an estimate $\hat{V}_{i,h_0+H}$ of the value function $V_{i,h_0+H} = \sum_{h=0}^{H} R_{i,h_0+h} = \log\left(Y^{(d)}_{i,h_0+H} / Y^{(d)}_{i,h_0}\right)$, and recover an estimate of the target :

$$\hat{Y}^{(d)}_{i,h_0+H} = \hat{Y}^{(d)}_{i,h_0} \exp(\hat{V}_{i,h_0+H}) \approx Y^{(d)}_{i,h_0} \exp(V_{i,h_0+H}) = Y^{(d)}_{i,h_0+H}$$

Finally, for days t within two time steps, we assume that the growth rate is constant between two time steps, equal to the next step growth rate and predict the outcome as

$$\begin{aligned}\hat{Y}_{i,t_0+h_0d+t} &= \hat{Y}^{(d)}_{i,h_0+\lfloor t/d \rfloor} \cdot \exp\left(\frac{t - d\lfloor t/d \rfloor}{d} \cdot \hat{R}_{h_0+\lfloor t/d \rfloor}\right) \\ &\approx Y^{(d)}_{i,h_0+\lfloor t/d \rfloor} \cdot \exp\left(\frac{t - d\lfloor t/d \rfloor}{d} \cdot R_{h_0+\lfloor t/d \rfloor}\right) = Y_{i,t_1+d\lfloor t/d \rfloor}\left(\frac{Y_{i,t_1+d\lfloor t/d \rfloor+d}}{Y_{i,t_1+d\lfloor t/d \rfloor}}\right)^{\frac{t-d\lfloor t/d \rfloor}{d}}\end{aligned}$$

where $t_1 = t_0 + h_0 d$.

Clustering of the outcomes. While the set of outcomes is populated by *log growth rates* observed over time across regions, the `MRL` method requires this set to be finite. As suggested in Bennouna et al. (2021), this additional characteristic could be achieved by defining a *distance threshold* ϵ, that will be used as a proxy for clustering the observed outcomes beforehand. This clustering creates batches of outcomes of the form $[R-\epsilon, R+\epsilon]$ that we consider to be similar outcomes. In other words, the algorithm does not distinguish between two outcomes in the same batch and identifies outcomes within ϵ. Hence, this introduces an additional error in the prediction of order $H\epsilon$, where H is the horizon of the prediction. Naturally, there is a trade-off captured. Taking ϵ too small increases the number of states of the constructed MDP (as a high precision in the prediction is required), creating a more complex structure to learn and generalize, while ϵ too large tends to oversimplify the learnt MDP and increases the discretization error, in which case the algorithm's prediction, within an error $H\epsilon$, would be of low accuracy.

Appendix B: Nearest-Neighbor Approach and Similarity-Weighted Time-Series

In this section, we discuss the convergence of the `KNN` model:

Let us define $m(x_{j\tau}) = E[Y_{j\tau}|X_{j\tau} = x_{j\tau}]$ the unknown regression function that we want to estimate from a compact subset $S_F \subset F$ to $\mathbb{R}$ where F is an infinite dimensional functional space.

The k-nearest neighbor estimator is defined as:

$$\hat{m}_{N,T}(x_{j\tau}) = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T} W_{it}(x_{j\tau}; X_{it})Y_{l\tau}}{\sum_{i=1}^{N}\sum_{t=1}^{T} W_{it}(x_{j\tau}; X_{it})} \tag{2}$$

where $N*T$ is the total number of observations, k is the number of nearest neighbors and $W_{it}(x_{j\tau}; X_{it})$ is the weight function usually taking the form of a kernel: $K\big(\frac{d(x,X_{it})}{h(x)}\big)$ for a distance metric d and a bandwidth h.

Let S_F be a compact subset of F, and $N_\delta(S_F)$ be the minimal number of open balls with radius δ in F which is necessary to cover S_F with centers $\chi_1, ..., \chi_{N_\delta(S_F)}$ respectively. In addition, let $\psi_{S_F}(\delta)$ be the Kolmogrov's δ-entropy of S_F. For all $\chi \in S_F$, denote $k(\chi) = \arg\min_{k\in\{1,2,...,N_\delta(S_F)\}} d(\chi, \chi_k)$,

$$\begin{aligned} s^2_{N*T,1} &= \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{N}\sum_{m=1}^{T} |Cov(Y_{i,t}u_{i,t}, Y_{j,m}u_{j,m})|, \quad s^2_{N*T,2} = \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{N}\sum_{m=1}^{T} |Cov(u'_{i,t}, u'_{j,m})| \\ s^2_{N*T,3} &= \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{N}\sum_{m=1}^{T} |Cov(u_{i,t}, u_{j,m})|, \qquad\qquad s^2_{N*T,4} = \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{N}\sum_{m=1}^{T} |Cov(u''_{i,t}, u''_{j,m})| \end{aligned} \tag{3}$$

where

$$\begin{aligned} u_{i,t} &= I_{B(\chi_{k(\chi)}, Ch_{N*T}+\delta)}(\chi_{i,t}) \text{ for some } C>0 \text{ and } 0 < h_{N*K} \to 0 \\ u'_{i,t} &= \frac{Y_{i,t}K\big(\frac{d(\chi_k,\chi_{i,t})}{h_{N*T}(\chi_k)}\big)}{EK\big(\frac{d(\chi_k,\chi_1)}{h_{N*T}(\chi_k)}\big)} - \frac{E\big(Y_{i,t}K\big(\frac{d(\chi_k,\chi_{i,t})}{h_{N*T}(\chi_k)}\big)\big)}{EK\big(\frac{d(\chi_k,\chi_1)}{h_{N*T}(\chi_k)}\big)} \\ u''_{i,t} &= \frac{K\big(\frac{d(\chi_k,\chi_{i,t})}{h_{N*T}(\chi_k)}\big)}{EK\big(\frac{d(\chi_k,\chi_1)}{h_{N*T}(\chi_k)}\big)} - \frac{EK\big(\frac{d(\chi_k,\chi_{i,t})}{h_{N*T}(\chi_k)}\big)}{EK\big(\frac{d(\chi_k,\chi_1)}{h_{N*T}(\chi_k)}\big)} \end{aligned} \tag{4}$$

We define

$$s^2_{N*T} = \max(s^2_{N*T,1}, s^2_{N*T,2}, s^2_{N*T,3}, s^2_{N*T,4}) \tag{5}$$

Let C, C', C_1, C_2 be some strictly positive generic constants. The assumptions under which the theorem holds are the following:

$$
\begin{aligned}
&(A1)\quad \forall \delta > 0, P(\chi \in B(\chi,\delta)) =: \varphi_\chi(\delta) > 0 \text{ and } \varphi_\chi(\cdot) \text{ is continuous and strictly increasing around } \\
&\quad 0 \text{ with } \varphi_\chi(0) = 0. \\
&(A2)\quad \exists \text{ function } \phi(\cdot) \geq 0, \text{a bounded function } f(\cdot) > 0, \alpha > 0 \text{ and } \tau > 0 \text{ such that} \\
&\quad (i)\quad \phi(0) = 0 \text{ and } \lim_{\delta \to 0} \phi(\delta) = 0. \\
&\quad (ii)\quad \lim_{\delta \to 0} (\phi(u\delta)/\phi(\delta)) = u^\alpha \text{ for } u > 0 \\
&\quad (iii)\quad \sup_{\chi \in S_F} |\varphi_\chi(\delta)/\phi(\delta) - f(\chi)| = \mathcal{O}(\delta^\tau), \text{ as } \delta \to 0. \\
&(A3)\quad \text{The kernel function } K(.) \text{ is a nonnegative function over its support } [0,1], \text{ and its} \\
&\quad \text{derivative } K'(.) \text{ exists on } [0,1] \text{ with } -\infty < C_1 < K'(t) < C_2 < 0 \text{ and } K(1) > 0 \quad \forall t \in [0,1]. \\
&(A4)\quad (i)\quad m(.) \text{ is a bounded Lipschitz operator of order } \beta \text{ on } S_F, \text{ that is, } \exists \beta > 0 \text{ such that} \\
&\quad \forall \chi_1, \chi_2 \in S_F, |m(\chi_1) - m(\chi_2)| \leq C d^\beta(\chi_1, \chi_2). \\
&\quad (ii)\quad \forall m \geq 2, E(|Y|^m | X = \chi) = \delta_m(\chi) < C \text{ with } \delta_m(\cdot) \text{ being continuous on } S_F. \\
&(A5)\quad \text{The Kolmogorov's } \delta - \text{entropy of } S_F \text{ satisfies } \sum_{n=1}^{\infty} e^{(1-\omega)\psi_{S_F}(\frac{\log n}{n})} < \infty \\
&\quad \text{for some large enough } \omega. \\
&(A6)\quad \exists \alpha > 1 \text{ and } p > 2 \text{ such that } s_{N*T}^{-((\alpha+1)p/(\alpha+p))} = o((N*T)^{-\theta}) \text{ for some large enough } \theta.
\end{aligned}
\tag{6}
$$

For the convergence of the $\hat{m}_{N,T}(x_{j\tau})$ in strong mixing time series data with a kernel as a weight function we will cite the following theorem.

THEOREM 2 **(see Ling et al. (2019))**. *Under the assumptions (A1)-(A6), if* $\lim_{N*T \to \infty} \frac{k}{N*T} = 0$, $\log^2 \frac{N*T}{k} < \psi_{S_F}(\frac{\log N*T}{N*T}) < \frac{k}{\log N*T}$ *and* $0 < C_1 < \frac{k}{\log^2 N*T} < C_2 < \infty$ *for* $N*T$ *large enough and* C_1,C_2 *some constants then*

$$
\sup_{x \in S_F} |\hat{m}_{N,T}(x) - m(x)| = \mathcal{O}_{a.co}\left(\phi^{-1}\left(\frac{k}{N*T}\right)^\beta + \sqrt{\frac{s_{N*T}^2 \psi_{S_F}(\frac{\log N*T}{N*T})}{(N*T)^2}} \right)
$$

Theorem 2 establishes uniform almost complete convergence of the estimator. The assumption stated in the theorem are quite usual for time series data. For those interested in this topic we recommend reading Ling et al. (2019) and the references within.

Appendix C: Deep Learning Approach

In the time-series variation of k-means, the similarity between two time series is measured by the DTW distance metric, and the cluster centroids are computed with respect to it. More specifically, given two time series $A = (a_0, ..., a_n)$ and $B = (b_0, ..., b_m)$ the DTW distance from A to B is formulated as the following optimization problem.

$$
DTW(A,B) = \text{minimize}_\pi \left(\sum_{(i,j)\in\pi} d(a_i, b_j)^2 \right)^{1/2}, \tag{7}
$$

with $\pi = [p_0, ..., \pi_K]$ being a path which is a list of index pairs such that for each $k \in \{0, 1, ..., K\}$, $\pi_k = (i_k, j_k)$, $i_k \in \{0, 1, ..., n-1\}$ and $j_k \in \{0, 1, ..., m-1\}$. Moreover, $\pi_0 = (0, 0)$, $\pi_K = (n-1, m-1)$, $i_{k-1} \leq i_k \leq i_{k-1} + 1$ and $j_{k-1} \leq j_k \leq j_{k-1} + 1$ where the distance function $d(\cdot, \cdot)$ in this case is the Euclidean distance. This method calculates an optimal match between the two sequences that has the minimal cost according to the used distance function.

Regarding Recurrent Neural Networks or RNNs, they are a category of neural networks that processes sequential data. RNNs model sequences of data so that each sample is assumed to be dependent on the previous one. Even though RNNs theoretically can process long sequences (also referred to as long term dependencies), in practice they do not perform well (see Sherstinsky (2020)). Moreover, they suffer from what is referred to as the gradient vanishing and exploding problem. LSTMs are designed to be a solution to those problems (Hochreiter (1998)). Additionally, their special architecture helps them remember and learn long term dependencies. LSTM models use special types of units that help them remember previous data and while simultaneously addressing gradient problems. Furthermore, they are capable of processing and predicting large time series.

In our system, each LSTM Network consists of two distinct layers. Each LSTM layer consists of multiple LSTM units connected sequentially. Each LSTM unit consists of a cell, an input gate, an output gate and a forget gate. The cell is the key component of the LSTM unit as it stores all the useful previous information. The LSTM can add and/or remove information to/from the cell via regulators called gates. By using specific activation functions, gates determine how much of each component of the unit goes through to the cell state. In order to formally describe the LSTM unit we introduce the following notation. Let $x_t \in \mathbb{R}^d$ be the input vector in the LSTM unit, $f_t \in \mathbb{R}^h$ the forget gate's activation vector, $i_t \in \mathbb{R}^h$ the input gate's activation vector, $o_t \in \mathbb{R}^h$ the output gate's activation vector, $h_t \in \mathbb{R}^h$ the output vector of the unit, $\tilde{c}_t \in \mathbb{R}^h$ the cell input activation vector, and $c_t \in \mathbb{R}^h$ the cell state vector.

The forget gate's activation vector is given by the following equation $f_t = \sigma(U_f h_{t-1} + W_f x_t + b_f)$, where σ is the sigmoid activation function and $U_f \in \mathbb{R}^{h \times h}, W_f \in \mathbb{R}^{h \times d}$ and $b_f \in \mathbb{R}^h$. This gate controls the self-loop weight (between 0 and 1) of the RNN. Intuitively, a value equal to 0 means that the LSTM decides to forget the previous unit's output, while a value equal to 1 means that the LSTM decided to keep completely the information from the previous unit.

At the same time, we calculate the input gate activation vector using the equation $i_t = \sigma(U_i h_{t-1} + W_i x_t + b_i)$, where σ is the sigmoid activation function, $U_i \in \mathbb{R}^{h \times h}, W_i \in \mathbb{R}^{h \times d}$, and $b_i \in \mathbb{R}^h$. The input gate activation vector is element-wise multiplied (Hadamard product) with the cell input activation vector, which is given through equation $\tilde{c}_t = \sigma(U_c h_{t-1} + W_c x_t + b_c)$, where σ is the hyperbolic tangent activation function, $U_c \in \mathbb{R}^{h \times h}, W_c \in \mathbb{R}^{h \times d}$, and $b_c \in \mathbb{R}^h$. The cell input activation function denotes the new candidate value for the cell and the input gate activation denotes the quantity of the new candidate value that will eventually be stored in the cell. Similar to what we described above, a value equal to 0 means that the LSTM decides to not take into consideration the new cell candidate value, while a value equal to 1 represents that the LSTM forwards completely the new candidate value to the cell.

The cell state vector is calculated using equation $c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$, where $\circ$ denotes the Hadamard product. The new cell state is given by summing the remaining, multiplied by f_t, information from the

previous cell, with the new candidate cell value multiplied by i_t. As before, this shows how much consideration we put into the candidate cell value.

Finally, the output value h_t is given by a filtered state of the cell state. As before, by computing the output's gate activation vector $o_t = \sigma(U_o h_{t-1} + W_o x_t + b_o)$, where σ is the sigmoid activation function, $U_f \in \mathbb{R}^{h \times h}, W_f \in \mathbb{R}^{h \times d}$, and $b_f \in \mathbb{R}^h$, we decide which part of the cell state will remain. Then, the cell state c_t passes through a hyperbolic tangent activation function, that scales the values between -1 and 1, and is element-wise multiplied with o_t, resulting in the output $h_t = o_t \circ \sigma(c_t)$, where σ is the hyperbolic tangent activation function. In Figure 2, a complete LSTM unit is depicted.

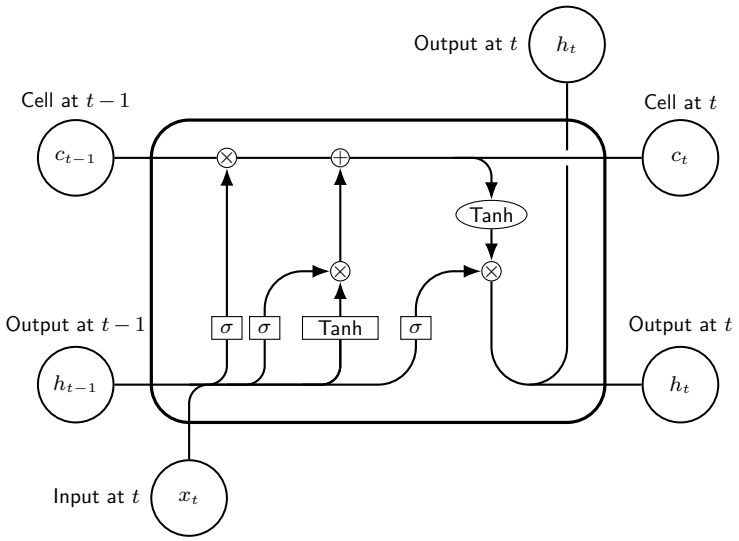

Figure 2 An LSTM unit.

The complete architecture of our system is depicted in the following figure.

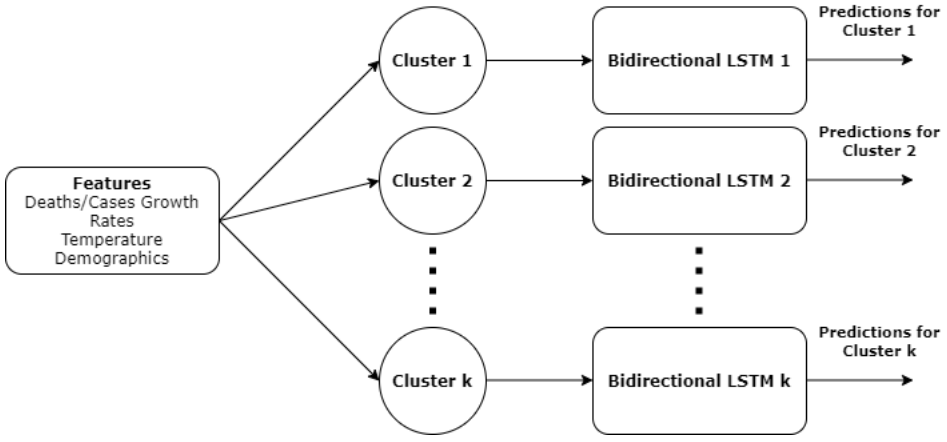

Figure 3 Architecture of the complete system.

Bounds on Prediction Accuracy

In this subsection, we establish a theoretical bound on the predictive accuracy of the `Bi-LSTM` network. To accomplish this we impose the following reasonable assumptions:

Assumption 1: The input data are bounded.

Assumption 2: The spectral norms of weight matrices are bounded. More specifically, $||W_f||_2 \leq B_{W_f}$, $||W_i||_2 \leq B_{W_i}$, $||W_c||_2 \leq B_{W_c}$, $||W_o||_2 \leq B_{W_o}$, $||U_f||_2 \leq B_{U_f}$, $||U_i||_2 \leq B_{U_i}$, $||U_c||_2 \leq B_{U_c}$, $||U_o||_2 \leq B_{U_o}$.

Assumption 3: The activation function of the output σ is Lipschitz continuous with parameter ρ and $\sigma(0) = 0$.

THEOREM 3 **(see Chen et al. (2019))**. *If Assumptions 1-3 hold, then for* $(x_t, z_t)_{t=1}^T$ *and* $S = \left\{(x_{i,t}, z_{i,t})_{t=1}^T, i = 1, ..., m\right\}$ *i.i.d. samples drawn from any underlying distribution over* $\mathbb{R}^{d_x \times T} \times \{1, .., K\}$*, with probability at least* $1 - \delta$ *over* S*, for every margin value* $\gamma > 0$ *and every* $f_t \in \mathcal{F}_{g,t}$ *for integer* $t \leq T$ *we have that:*

$$\mathbb{P}[\hat{z} \neq z] \leq \hat{\mathcal{R}}_\gamma(f_t) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}} + \mathcal{O}\left(\frac{d\rho B_V \min\{\sqrt{d}, B_{W_c} B_x \frac{\beta^t - 1}{\beta - 1}\}\sqrt{\log(\frac{\theta^t - 1}{\theta - 1} d\sqrt{m})}}{\sqrt{m}\gamma}\right), \tag{8}$$

where $\hat{z}$ *is the estimation of the LSTM,* $\hat{\mathcal{R}}_\gamma(f_t)$ *is the empirical risk,* $\mathcal{F}_{g,t} = \{(X_t, z_t) \mapsto g(f_t(X_t), z_t) : f_t \in \mathcal{F}_t\}$*, where* $\mathcal{F}_t$ *is the class of mappings from the first* t *inputs to the* $t-th$ *output and* g *the loss function,* $d_x = m$ *is the number of features,* K *is the total number of labels,* d *is the maximum dimension of the matrices,* B_V *is the bound on the spectral norm of the matrix applied to the output* h_t*, and finally,* B_x *is the bound of the second norm of each data point,* $\beta = \max\{||f_j||_\infty + B_{U_c}||i_j||_\infty||o_j||_\infty\}$ *and* $\theta = \beta + B_{U_f} + B_{U_i} + B_{U_o}$.

This result follows from Chen et al. (2019). The bound it establishes holds asymptotically for regression and is of the order of $\mathcal{O}(\frac{\sqrt{\log \frac{1}{\delta}}\gamma + d\min\{\sqrt{d}, \beta^t\}\sqrt{\log(\theta^t d\sqrt{m})}}{\sqrt{m}\gamma})$. Since, the bidirectional LSTM consists of two independent LSTMs the generalization bounds hold asymptotically also for the Bi-directional LSTM.

Appendix D: Epidemiology Approach

We begin by briefly describing the original SEIRD model and then build on it to formulate the multi-peak `C-SEIRD` model. The SEIRD model is a compartmental epidemiology model. It assumes that the total population is of size N and can be broken into five sub-populations: susceptible (S), exposed (E), infected (I), recovered (R) and deceased (D). Individuals move through these sub-populations, and the aggregate of this movement is described by the following set of differential equations:

$$\frac{dS}{dt} = -\frac{\beta SI}{N}, \quad \frac{dE}{dt} = \frac{\beta SI}{N} - \frac{E}{\alpha}, \quad \frac{dI}{dt} = \frac{E}{\alpha} - (\gamma + \mu)I, \quad \frac{dR}{dt} = \gamma I, \quad \frac{dD}{dt} = \mu I \tag{9}$$

In these equations β represents the infection rate of the disease, α represents the average incubation time (i.e., how long it takes for an individual to move from getting the disease to being contagious), γ denotes the recovery rate of infected individuals, and μ denotes the mortality rate.

Nevertheless, this formulation assumes that the parameters are static. If this were the case, then any area with COVID-19 would have only experienced a single wave (peak) of the disease before the virus had time to mutate or people had enough time to lose their immunity. However the multiple waves of COVID-19 seen across the United States demonstrate that the parameters are not static. Instead, population behavior and environmental conditions have driven multiple waves, leading to time-dependent parameters.

References

Bennouna, A., Pachamanova, D., Perakis, G., and Skali Lami, O. (2021). Learning the minimal representation of a dynamic system from transition data. *https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3785547*.

Chen, M., Li, X., and Zhao, T. (2019). On generalization bounds of a family of recurrent neural networks. *CoRR*, abs/1910.12947.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, 6:107–116.

Ling, N., Meng, S., and Vieu, P. (2019). Uniform consistency rate of k nn regression estimation for functional time series data. *Journal of Nonparametric Statistics*, 31(2):451–468.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306.