MACHINE LEARNING FOR EPIDEMIOLOGICAL MODELS

E. Meloni, A. Zugarini, A. Panizza, A. Betti, M. Gori
September 24, 2020

SAILab, University of Siena
1. SIR Models

2. The data

3. The learning problem

4. Regularization

5. Fitting the data
SIR Models
WHY LEARN EPIDEMIC MODELS?

During 2020, the spread of COVID-19 infection affected the whole world.

Some countries resorted to non-pharmaceutical interventions of various degrees to control the spread of the virus, such as lockdowns or imposing the use of masks.

Many countries collected vast amounts of detailed data about the evolution of the epidemic.

Understanding the evolution of the epidemic from the data can help us understand the efficacy of this interventions and predict the spread of the illness if no action is taken.
WHY LEARN EPIDEMIC MODELS?

Deterministic epidemic models, such as SIR, usually model the evolution through some constant parameters, such as infection rate.

Constant parameters are ill-suited to model a dynamic phenomenon where the spread is counter-acted by the intervention of national authorities.

Time-varying parameters are better suited, but the parameter space becomes quickly too big to be manually fine-tuned where analytical solutions do not exist.

A Machine Learning approach can be used to explore the parameter space and find interesting and useful models.
A classical **deterministic** approach to model an epidemic is the *susceptible, infectious, removed* (SIR) model.

A population of $N$ individuals is divided into

- $x(t) = (# \text{ of susceptible at time } t)$
- $y(t) = (# \text{ of infectious at time } t)$
- $z(t) = (# \text{ of removed at time } t)$

We assume that the population remains **constant**, therefore:

$$x(t) + y(t) + z(t) = N.$$
THE MODEL

The dynamics of the epidemics then can be described by the following system of ODEs\(^1\):

\[
\begin{align*}
x'(t) &= -\beta x(t)y(t) \\
y'(t) &= \beta x(t)y(t) - \gamma y(t) \\
z'(t) &= \gamma y(t)
\end{align*}
\]

- \(\beta > 0 \rightarrow \) infection rate
- \(\gamma > 0 \rightarrow \) removal rate
- \(R_0 := \beta/\gamma \rightarrow \) reproduction number

A more sophisticated epidemiological model is **SIDARTHE**\(^2\), where we add **more compartments** to better describe the stages of infection. (ODEs can be found in the Appendix)

Compartments:

- **S**: Susceptible
- **I**: Asymptomatic undetected infected individuals
- **D**: Asymptomatic detected infected individuals
- **A**: Symptomatic undetected infected individuals
- **R**: Symptomatic detected infected individuals
- **T**: Acutely symptomatic infected individuals
- **H**: Healed individuals
- **E**: Deceased individuals

An important difference is that we differentiate between detected and undetected individuals.
The epidemiological parameters are usually considered **constant in time**.

This prevents the model from describing changes due to **non-pharmaceutical interventions** (lockdown, etc...).

We want to extend the model by allowing **daily changes** to the parameters values.

\[
\beta \rightarrow \beta(t) \quad \tau \rightarrow \tau(t)
\]

In this way we can monitor how parameters (i.e. infection rate) **change** during the epidemic evolution.
The data
We pre-processed the data\textsuperscript{3} to obtain 5 time-series which map to 5 SIDARTHE compartments:

- $\overline{D}$: detected asymptomatic individuals
- $\overline{R}$: detected symptomatic individuals
- $\overline{T}$: acutely symptomatic individuals
- $\overline{H}_d$: portion of healed individuals that were previously detected
- $\overline{E}$: deceased individuals

We do not have access to data about the \textbf{undetected} infected individuals (of course).

\textsuperscript{3}You can find the \textbf{data} following this link: https://github.com/pcm-dpc/COVID-19
All time-series were split into three sets:

- **Train Set** (size $T$): $0 \leq t \leq T$
- **Validation Set** (size $V$): $T \leq t \leq T + V$
- **Test Set** (size $D$): $T + V \leq t \leq T + V + D$

As usual, the training set will be used to fit the model, the validation set will be used to evaluate over-fitting and the test set will be used for the final evaluation of the model.
The learning problem
We call $u(t)$ the **concatenation** of all parameters at time $t$.

$$u(t) = (\alpha(t), \beta(t)...)$$

**Problem**

**Learning** $u(t)$ from **supervisions** on the number of people in the **detected** compartments, i.e. $\overline{D}$, $\overline{R}$, $\overline{T}$, $\overline{H_d}$ and $\overline{E}$. 
Given the supervisions $\bar{D}$, $\bar{R}$, $\bar{T}$, $\bar{H}_d$ and $\bar{E}$

Given $\hat{\bar{D}}^u$, $\hat{\bar{R}}^u$, $\hat{\bar{T}}^u$, $\hat{\bar{H}}^u_d$ and $\hat{\bar{E}}^u$ the solutions of SIDARTHE equations given by parameters $u(t)$.

We want to find the set of parameters $u(t)$ which minimizes the MSE of the prediction against the targets on samples for $t = 0 \rightarrow T$. 
The learning is done through **Gradient Descent**.

The SIDARTHE equations are computed through **numeric integration**, using **Heun** method and a fixed time step.

To obtain the gradient of parameters through the integration, we have implemented the Heun method in **PyTorch** to leverage its powerful autograd framework.

**Sidenote**: there are strong similarities with **Neural ODEs**.
Regularization
In case of overfitting, the learned parameters could result in a discontinuous and wrinkled function of time.

To avoid this problem and reduce overfitting, we augment the loss function with a regularization term $R(u)$. The penalty $R(u)$ is higher when the parameters are discontinuous and enforces a smooth function.
Since the solution is computed through integration, the value of parameters at initial time steps have a **bigger impact** on the solution w.r.t. values at final time steps.

This can be seen in the gradient of each parameter: for $t \to T$, gradients **approach** 0.

Hence, we augmented the Gradient Descent update rule with a **momentum** term, which manages to **distribute** the gradient throughout the $u$ function.

**Important note**: the commonly known momentum carries the gradient over subsequent **epochs**; this momentum term carries the gradient over subsequent **time steps** of the parameters $u(t)$
$D$, $R$, $T$, $H_d$ and $E$ have very different scales of values.

E.g. $\max(H_d) \approx 200'000$ and $\max(T) \approx 4'000$

This could cause the learning to prefer a target at the expense of another one.

We weight the loss components for each target based on its scale relative to the other targets.
Fitting the data
ACUTELY SYMPTOMATIC & DECEASED
DETECTED HEALED INDIVIDUALS

H_DETECTED - train/validation/test

- Prediction (train)
- Target (train)
- Prediction (val)
- Target (val)
- Prediction (test)
- Target (test)
PARAMETERS

Infection Rates

Detection Rates

Death Rates

Symptoms Development Rates
Thank you for listening!
References


\[
\begin{align*}
\dot{S}(t) &= -S(t)(\alpha I(t) + \beta D(t) + \gamma A(t) + \delta R(t)); \\
\dot{I}(t) &= S(t)(\alpha I(t) + \beta D(t) + \gamma A(t) + \delta R(t)) - (\varepsilon + \zeta + \lambda)I(t); \\
\dot{D}(t) &= \varepsilon I(t) - (\eta + \rho)D(t); \\
\dot{A}(t) &= \zeta I(t) - (\theta + \mu + \kappa + \phi)A(t); \\
\dot{R}(t) &= \eta D(t) + \theta A(t) - (\nu + \xi + \chi)R(t); \\
\dot{T}(t) &= \mu A(t) + \nu R(t) - (\sigma + \tau)T(t); \\
\dot{H}(t) &= \lambda I(t) + \rho D(t) + \kappa A(t) + \xi R(t) + \sigma T(t); \\
\dot{E}(t) &= \phi A(t) + \chi R(t) + \tau T(t)
\end{align*}
\]
The loss function $F(u)$ has 5 components, one for each target. Each target is weighted by the normalizing weight $W_x$, which takes into account the difference in scale relatively to other targets.

Reminder: $\hat{X}$ is the target, $X^u$ is the solution computed by the model using parameters $u(t)$

$$F(u) = \frac{1}{T} \sum_{t=0}^{T} \frac{W_D}{2} (D(t) - \hat{D}^u(t))^2 + \frac{W_R}{2} (R(t) - \hat{R}^u(t))^2 + \frac{W_T}{2} (T(t) - \hat{T}^u(t))^2 + \frac{W_H}{2} (H(t) - \hat{H}^u_d(t))^2 + \frac{W_E}{2} (E(t) - \hat{E}^u(t))^2$$
The smoothness regularization term depends on the derivative of the parameters.

It is also weighted with the hyper-parameter $W_R$ to better tune the regularizing effect of the term.

$$ R(u) = \frac{W_R}{T} \sum_{t=0}^{T} \frac{\dot{u}(t)^2}{2} $$
The momentum term is added to the well-known update rule of Gradient Descent.

\[
 u_{t+1}^k = \begin{cases} 
 u_t^k - \nabla_t f(u^k) & \text{if } t = 0 \\
 u_t^k - \nabla_t f(u^k) + \mu_t(u_{t-1}^{k+1} - u_{t-1}^k) & \text{if } t > 0 
\end{cases}
\]

Where \( k \) is the epoch number, \( \nabla_t f(u^k) \) is the gradient of parameter \( u \) at time \( t \) and epoch \( k \), \( \mu(t) = \text{sigmoid}(mt) \) and \( m \) is a hyper-parameter.
SAILab: https://sailab.diism.unisi.it/

Code: https://github.com/sailab-code/learning-sidarthe

Data: https://github.com/pcm-dpc/COVID-19

Speaker webpage: https://enricomeloni.github.io