Forecasters Description

David Weber

October 23, 2024

Contents

This gives a more mathematical description of the various forecasters defined in this repository

1 Flu test case

Makes all historical data from ILI+, nhsn, and flusurv+ available, all as rows, with a source column to distinguish the origin for forecasters to filter on, and an `agg_level` column to distinguish `states` from `hhs_regions`; nhsn data has also been included at the aggregated to `hhs_region` level. ILI+ and flusurv+ mean that each has been adjusted so that the total for the season matches nhsn's total. Flusurv is taken from epidata, but ili+ was constructed by Evan and given to Richard. Currently looking to train on the 2023 season, so October 2023 through late April 2024.

2 `Flusion`

Note that this is a description of the flusion-like model, rather than the one written up in that paper. This is definitely the most complicated forecaster, it adds many non-ar components. First, on a per geo per group basis, each signal is transformed via:

$$x' = (x + 0.01)^{\frac{1}{4}}$$

$$\bar{x} = \frac{x' - \text{med}(x')}{\text{quant}(x', 0.95) - \text{quant}(x', 0.05)}$$

This differs from the data whitening used in the flusion paper in two ways: the first is subtracting the median, rather than the mean, and the second is scaling by the difference between the 5th and 95th quantiles, rather than just the 95th quantile. This is more closely in line with the RobustScaler from scikit-learn (using a much wider quantile than the default settings there). Neither we nor they have tested/examined whether a robust scaler is more appropriate than say a mean+std scaler for this case.

The model is roughly (dropping coefficients, and ignoring subtleties in the way different engines handle different variables)

$$\bar{x}_{t+k} = \text{ar}(\bar{x}) + t_{season} + p + d + \chi_{geo} + \chi_{source} + \left\langle \bar{x}_{t-k} \right\rangle_{k=0:1} + \left\langle \bar{x}_{t-k} \right\rangle_{t=0:3} + \text{de}(\bar{x})$$

Where:

- $\text{ar}(x)$ represents a typical and configurable AR process
- t_{season} is the week of the season (starting at 1 for epiweek 40, going to 52 at epiweek 39 of the next year)
- $t_{year} - 52$ gives the time in weeks until the last week of the year (Christmas + new years)
- p is the population
- d is the population density (not in the original flusion model)
- χ_{geo} is a one-hot indicator of the `geo_value` (so it actually adds # geo binary variables). This also encodes the $\text{geo}_{\text{scale}}$ (hhs region vs state).
- χ_{source} is a one-hot indicator of the `source` of the data. This is related to the scale but in a weighted rather than direct manner.

- $\langle x_{t-k} \rangle_{k=0:j}$ denoting the mean for the j values before present (so the previous 2 and 4 weeks)
- de is a collection of variables using some method of estimating the first and second derivatives. The method used in the original paper is a rather ad-hoc collection of linear and quadratic regressions. Initially we tried to use this as well via `get_poly_coefs`, but that was quite slow. Instead we are currently using the "$\text{rel}_{\text{change}}$" method of `step_growth_rate`, applied iteratively to get higher order derivatives. For the first derivative, we use three weeks, so explicitly, this is $d_t = \frac{1}{21}\left(\frac{x_t + x_{t-1}}{x_{t-2} + x_{t-1}} - 1\right)$. Similarly, we estimate the second derivative using the relative change in this d_t over 2 weeks, which is $\frac{1}{14}\left(\frac{d_t}{d_{t-1}} - 1\right)$.

At the moment this is only set up to use random forests as implemented by grf, as I believe its the only quantile model we have that introduces regularization.

I have a test set up to drop the state dummy variables, the source dummy variables, and the growth rates.

3 no_recent_outcome

A flusion-adjacent model pared down to handle the case of not having the target as a predictor.

$$\bar{x}_{t+k} = f(t_{season}) + p + d + \langle y_{t-k} \rangle_{k=0:1} + \langle y_{t-k} \rangle_{t=0:3}$$

y here is any exogenous variables; initially this will be empty, as nssp is missing some states, so we will have to rewrite these models to handle missing geos (most likely by having a separate model for the case when an exogenous variable is missing).

f is either the identity or 2 sine terms, defined so that the first has half a period during the season, and is zero after it, while the second is one period over the season, with zero after:

```
library(tidyverse)
res <- tibble(season_week = 1:52)
res %>% mutate(
          first_term = sinpi((pmin(season_week, 36) - 1)/35),
          second_term = sinpi(2*(pmin(season_week, 36) - 1)/35)
               ) %>%
  pivot_longer(cols = c("first_term", "second_term")) %>%
  ggplot(aes(x=season_week, y = value, color = name)) +geom_line()
```

I am also going to test having and dropping the robust scaling, the population p, and the density d (and using one but not the other).

4 `scaled_pop`

A simple model, which predicts using

$$x_{t+k} = ar(x)$$

where x can be scaled according to each state's population. I am going to try this both with/without scaling, and with/without the extra data sources (ILI+ and Flusurv), though because they're rates, the population scaling is likely to be unhelpful.

5 `smooth_scaled`

This is part of the covid forecaster we used last year, that performed approximately as well as any of the others. It takes the form:

$$x_{t+k} = ar(x) + \sum_{i=i_1,\ldots i_n} \left\langle x_{t-k} \right\rangle_{k=0:i} + \left\langle \left(x_{t-k} - \left\langle x_{t-k} \right\rangle_{k=0:j_0}\right)^2 \right\rangle_{k=0:j_1}$$

where x may or may not be population scaled. where $\left\langle x_{t-k} \right\rangle_{k=0:i}$ is just the mean over the last i weeks. While its set up to include multiple means, over multiple sources, at the moment we don't. The second term is a standard deviation of sorts, where the mean isn't necessarily over the same interval as the standard deviation (typically shorter). In practice, the mean i is just 7, the standard deviation mean j_0 is 14, and the standard deviation j_1 is 28.

Like $\text{scaled}_{\text{pop}}$, we're testing this both with and without population scaling, and with and without the extra data.

6 `flateline_fc`

This is just exactly the flatline forecaster from epipredict.