SurvSuperLearnerAlgorithm

ES

February 2023

1 Setup and Initialization

1.1 Data

We have time, event as observed from $i = 1$ to N, $N \times 1$ vectors. The observed data matrix of predictors is $\mathbf{X}$, of nrow N (number of predictors doesn't matter here).

Let k be the number of candidate algorithms (base learners).

1.2 Epsilon (ϵ)

Take unique values of observed time, sort them smaller to larger, take lagged differences $(t_{i+1} - t_i)$, and take the minimum of this.

$$\text{min(diff(sort(unique(time))))} \tag{ϵ}$$

1.3 Time Grid

Set up a sequence, going from min(time) to max(time), of size 250 (arbitrary, this is what is default in the algorithm)

$$\text{seq(from=min(time),max(time),size=250)} \tag{t.grid}$$

1.4 Long Form Event and Censoring

These are the LONG FORM predictions from base learners of the event and censoring values, for the TIME GRID.

They are matrices of size $(N * 250) \times k$ where 250 is the length of the time grid mentioned earlier.

Call these "event.Z.long" and "cens.Z.long".

1.5 "Observed" Event and Censoring

"event.Z.obs" and "cens.Z.obs" are the base learner algorithm predictions, but for the OBSERVED TIME instead of the grid. These are computed via interpolation with x as the time grid, and y as the predictions for the time grid, and "xout" as the observed times $t_1, \cdots, t_N$.

They are both matrices of dimensions $N \times k$.

1.6 Long Form time/Event/ObsWeights and Time Grid

Self explanatory time, event, weights and time grid all in long form:

rep(time,250)
rep(event,250)
rep(ObsWeights,250)
rep(t.grid,N)

All vectors of length $N * 250$

1.7 Initial Censoring Fit

By default; random forest fit (this is changable via the "control" parameter) onto observed time (not long form), and 1-event (so the censoring mechanism), with NEW TIMES: time-ϵ.

Predictions from this fit:

$$\text{preds} = \begin{array}{cc} & \begin{array}{cccc} t_1 - \epsilon & t_2 - \epsilon & \cdots & t_N - \epsilon \end{array} \\ \begin{array}{c} i=1 \\ i=2 \\ \vdots \\ i=N \end{array} & \begin{pmatrix} d_1 & . & . & . \\ . & d_2 & . & . \\ & & & \\ . & . & . & d_N \end{pmatrix} \end{array}$$

Take the DIAGONAL, so $t_1 - \epsilon$ for $i = 1, \cdots, t_N - \epsilon$ for $i = N$, so that each prediction is for the observed time for the corresponding subject. This would be a $N \times 1$ vector.

Now long form this, so it is a vector of length $(N*250)$. Call this "obs.cens.vals", or "observed censoring values".

This initializes the censoring values.

1.8 Initial Event

To initialize event values, we the function ComputeCoef function (explained in the next section) once, with inputs:

time = long form observed time $(N * 250 \times 1)$, event = long form observed events $(N * 250 \times 1)$, t.vals = long form time grid $(N * 250 \times 1)$, cens.vals =

obs.cens.vals from the previous subsection ($N*250\times1$), and preds = event.Z.long $(N*250)\times k$.

This results in a vector of coefficients, S.coef, of length k.

To initialize, take event.Z.obs ($N\times k$), multiply it by the coefficients we just found, S.coef ($k\times1$) to get a vector of length $N*1$.

Now, long form this (replicate 250 times) to get a size $N*250$ vector. Call this "obs.event.vals".

This initializes the event values.

2 Main Loop

WHILE (algorithm has not converged), do the following:

2.1 STEP 1: STORE OLD "OBSERVED" VALUES

First, save the previous values of "obs.cens.vals" and "obs.event.vals" that we initialized previously, as "obs.cens.vals.old" and "obs.event.vals.old":

$$\text{obs.cens.vals.old} = \text{obs.cens.vals}$$
$$\text{obs.event.vals.old} = \text{obs.event.vals}$$

2.2 STEP 2: UPDATE "obs.cens.vals":

Update the obs.cens.vals by using the compute coef function on the CENSORING:

G.coef = ComputeCoef(time = time.long,event = 1-event.long, t.vals = tgrid.long,cens.vals = obs.event.vals,preds = cens.Z.long)

take: cens.Z.obs ($N\times k$) * G.coef ($k\times1$)

replicate 250 times to get the long form, the new "obs.cens.vals", vector of length $N*250$.

2.3 STEP 3: UPDATE "obs.event.vals":

Update the obs.event.vals by using the compute coef function on the EVENT:

S.coef = ComputeCoef(time = time.long,event = event.long, t.vals = tgrid.long,cens.vals = obs.cens.vals,preds = event.Z.long)

take: event.Z.obs ($N\times k$) * S.coef ($k\times1$)

replicate 250 times to get the long form, the new "obs.event.vals", vector of length $N*250$.

NOTE that in both these depend on one another; the G.coef depends on the current values of "obs.event.vals", and S.coef depends on "obs.cens.vals", both of which are updated at each iteration of the loop.

2.4 STEP 4: ESTABLISH CONVERGENCE CRITERION

Let:

$$\delta\text{cens} =_{max} |obs.cens.vals - obs.cens.vals.old| \tag{δ cens}$$

$$\delta\text{event} =_{max} |obs.event.vals - obs.event.vals.old| \tag{δ event}$$

2.5 STEP 5: HALT/CONTINUE

if δ cens + δ event $<$ 1e-5, the algorithm has converged.

If not, continue until the max number of iterations is reached.

3 One Step Optimization (ComputeCoef function)

So what does ComputeCoef do?

3.1 Arguments

Time , Event, Time Grid, Censoring Values, Predictions, Observation Weights, ALL IN LONG FORM as discussed in the "setup" section.

- Time: rep(time,250), $(N * 250) \times 1$
- Event: rep(event,250), $(N * 250) \times 1$
- Time Grid: rep(t.grid,N), $(N * 250) \times 1$
- Censoring Values: updated at each step, $(N * 250) \times 1$, always for opposite event; if for event then it's "obs.cens.vals", if for censoring, it's "obs.event.vals"
- Predictions: if for event, then "event.Z.long" ($(N * 250) \times k$), if for censoring, then "cens.Z.long" ($(N * 250) \times k$)
- Observation Weights: rep(ObsWeights,250), $(N * 250) \times 1$

3.2 Optimization

Now, let:

$$A = \sqrt{ObservationWeights} * Predictions$$

and

$$B = \sqrt{ObservationWeights} * \left\{ 1 - \left\{ \frac{\mathbf{I}(time < TimeGrid) \times event}{CensoringValues} \right\} \right\}$$

Use NNLS to solve $_{argmin(x)} \|Ax - b\|_2$ where $\|a\|_2 = \sqrt{x * x}$

So we're minimizing $\|Ax - b\|_2 = \sqrt{(Ax - b)(Ax - b)} = \sqrt{(Ax - b)^2}$.

Note the multiplications are item by item; so A has the same dimensions as the Predictions; $(N * 250) \times k$ and B has the same dimensions as long form event, $(N * 250) \times 1$

The resulting x is a vector of parameters of length k, asking "what is the optimal vector of parameters x that when multiplying A, get us closest to B. Or like "fitting A onto B"

The long form of event, predictions, observation weights and time and time grid never change. The only thing differing are the CENSORING VALUES, denominator of the B term.