

# Cox regression: Estimation

Patrick Breheny

October 29

# Introduction

- In our last lecture, we introduced the Cox partial likelihood; today, we will go over how to solve for  $\hat{\beta}$ , the maximum (partial) likelihood estimator
- As in previous models, this will require working out the score vector and Hessian matrix and applying an iterative Newton procedure
- On a superficial level this procedure is similar to our other regression models, but the details are quite different: although the observations are independent, we can no longer treat the partial likelihood contributions from each observation in isolation

# Partial likelihood; at-risk indicator

- Recall the Cox partial likelihood (PL):

$$L(\boldsymbol{\beta}) = \prod_j \frac{\exp(\mathbf{x}_j^T \boldsymbol{\beta})}{\sum_{i \in R(t_j)} \exp(\mathbf{x}_i^T \boldsymbol{\beta})},$$

where  $j$  indexes the observed failure times (for the sake of today's lecture, we will assume they are unique) and  $R(t)$  is the set of observations at risk at time  $t$

- The denominator in the expression above is also sometimes written as

$$\sum_{i=1}^n Y_i(t_j) \exp(\mathbf{x}_i^T \boldsymbol{\beta}),$$

where  $Y_i(t)$  is an *at-risk indicator*, equal to 1 if subject  $i$  is at risk at time  $t$ , and 0 otherwise

## Cox PL in terms of individual weights

- As an alternative, it is often convenient to express the likelihood as a product of terms for each time on study (as opposed to only the failure times)
- To simplify the expression, let  $w_j = \exp(\mathbf{x}_j^T \boldsymbol{\beta})$ ; the Cox partial likelihood can now be written as

$$L(\boldsymbol{\beta}) = \prod_{j=1}^n \left\{ \frac{w_j}{\sum_{R_j} w_i} \right\}^{d_j};$$

note that there are now  $n$  terms in the product (one for each individual, as opposed to one for each failure time)

- Expressing the partial likelihood in this way emphasizes the fact that the model assigns weights  $w_i$  to the likelihood that individual  $i$  will fail relative to the other subjects at risk

## Comments

- Note that the  $d_j$  exponent ensures that only the observations at which a failure is observed contribute to the likelihood
- However, because each subject affects the total hazard  $\sum_{R_j} w_i$  over all the failure times at which they are in the risk set, the contribution that subject  $i$  makes to the likelihood is not limited to the  $i$ th term in the product
- Because this sum will appear many times in our derivations today, I will denote it  $W_j$ :

$$W_j = \sum_{i \in R(t_j)} w_i,$$

where  $W_j$  represents the total hazard for all subjects at risk for the time at which subject  $j$  fails

## Failure probabilities

- The relative probability of failure for subject  $i$  is given by  $w_i$ ; let us denote the absolute probability of failure for subject  $i$  at time  $t_j$  as  $\pi_{ij}$ :

$$\pi_{ij} = Y_i(t_j) \frac{w_i}{W_j}$$

- Note, of course, that this probability is absolute only in the conditional sense, given that a failure occurs at time  $t_j$

# Log-likelihood

- The (partial) log-likelihood is therefore

$$\begin{aligned}\ell &= \sum_j d_j \log w_j - \sum_j d_j \log W_j \\ &= \sum_j d_j \eta_j - \sum_j d_j \log W_j\end{aligned}$$

- As we begin to take derivatives, keep in mind that the  $W_j$  term contains many  $\eta$  terms in addition to  $\eta_j$

# Score

- Solving for  $\hat{\beta}$  involves deriving the score equations and setting them equal to zero
- Let us begin by evaluating the partial derivative of the likelihood with respect to the  $i$ th linear predictor:

$$\frac{\partial \ell}{\partial \eta_i} = d_i - \sum_j \pi_{ij} d_j$$

- Thus, we can write the derivative of the log-likelihood with respect to the vector of linear predictors as

$$\nabla_{\boldsymbol{\eta}} \ell = \mathbf{d} - \mathbf{P}\mathbf{d},$$

where  $\mathbf{P}$  is an  $n \times n$  matrix with  $\pi_{ij}$  as its  $i, j$ th element



## Score (continued)

- As we have seen before, by the chain rule the score with respect to  $\beta$  is therefore

$$\mathbf{u}(\beta) = \mathbf{X}^T(\mathbf{d} - \mathbf{P}\mathbf{d})$$

- Alternatively, we can express the score equations as

$$\sum_{j:d_j=1} (\mathbf{x}_j - \mathbb{E}_j \mathbf{x}) = \mathbf{0},$$

where  $\mathbb{E}_j \mathbf{x} = \sum_i \mathbf{x}_i \pi_{ij}$  can be thought of as the expected value of the covariate vector at the  $j$ th failure time given the probability distribution implied by the model

# Hessian

- The score, of course, is nonlinear in  $\beta$ , meaning that we will have to apply a Taylor series expansion in order to solve it
- This, in turn, involves finding second derivatives: i.e., the Hessian matrix
- Let us start with the diagonal elements (with respect to the linear predictors):

$$\frac{\partial^2 \ell}{\partial \eta_i^2} = - \sum_j d_j \pi_{ij} (1 - \pi_{ij})$$

- Similarly,

$$\frac{\partial^2 \ell}{\partial \eta_i \partial \eta_k} = \sum_j d_j \pi_{ij} \pi_{kj}$$

# Hessian (continued)

- Again, applying the chain rule we obtain the Hessian with respect to  $\beta$ :

$$\mathbf{H}(\beta) = -\mathbf{X}^T \mathbf{W} \mathbf{X},$$

where  $\mathbf{W}$  denotes the (non-diagonal) matrix whose terms are given on the previous slide, with signs reversed (note that  $\mathbf{W}$  is unrelated to  $W_i$ ; my apologies if the notation is confusing)

- Alternatively, one can express the Hessian as

$$-\mathbf{H}(\beta) = \sum_{j:d_j=1} \sum_i \pi_{ij} (\mathbf{x}_i - \mathbb{E}_j \mathbf{x}) (\mathbf{x}_i - \mathbb{E}_j \mathbf{x})^T$$

# Newton algorithm

- As we have seen previously with the exponential and Weibull regression models, Newton's method is an effective, efficient iterative procedure that converges to the MLE (usually)
- For Cox regression, the Newton update is given by

$$\hat{\beta}_{(m+1)} = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{d} - \mathbf{P} \mathbf{d}) + \hat{\beta}_{(m)},$$

where  $\mathbf{W}$  and  $\mathbf{P}$  are evaluated at  $\hat{\beta}_{(m)}$ , the current value of the regression coefficients

## Crude R code

```
for (i in 1:20) {  
  eta <- X %*% b  
  haz <- as.numeric(exp(eta))      # w[i]  
  rsk <- rev(cumsum(rev(haz)))     # W[i]  
  P <- outer(haz, rsk, '/')  
  P[upper.tri(P)] <- 0  
  W <- -P %*% diag(d) %*% t(P)  
  diag(W) <- diag(P %*% diag(d) %*% t(1-P))  
  b <- solve(t(X)%*%W%*% X) %*% t(X) %*% (d - P%*%d) + b  
}
```

The above code assumes that the data has been sorted by time on study, and assumes no ties are present

## Comments

- The code on the previous slide is crude for several reasons:
  - It could be faster/more efficient
  - It doesn't check for convergence
  - It can occasionally fail to converge, because it doesn't implement step-halving when needed
- You are tasked with addressing the last two shortcomings on your next homework assignment

## Examples: pbc data

Some examples for how well Newton's method works on the pbc data:

- Model contains `trt`, `stage`, and `hepato`: Converges in 4 iterations
- Model contains `trt`, `stage`, `hepato`, and `bili`: Fails to converge
- Model contains `trt`, `stage`, `hepato`, and `bili`, but we employ step-halving: Converges in  $\sim 20$  iterations

## Conditional step-halving

- The `survival` package, however, can fit the Cox model with `trt`, `stage`, `hepato`, and `bili` in just 6 iterations ... how does it do that?
- The fundamental tradeoff here is between stability and speed: step-halving slows down convergence (intentionally!), but provides stability
- It would be desirable to use Newton's method as a default, but have some sort of check in place that uses step-halving when problems arise



## Likelihood checking

- It turns out that this is fairly straightforward to accomplish
- Let  $\tilde{\beta}$  denote the Newton update, and consider the following procedure:
  - (1) Calculate  $\ell(\hat{\beta}_{(m)})$
  - (2) Calculate  $\ell(\tilde{\beta})$
  - (3) If  $\ell(\tilde{\beta}) > \ell(\hat{\beta}_{(m)})$ , then  $\hat{\beta}_{(m+1)} \leftarrow \tilde{\beta}$ ; otherwise,  

$$\hat{\beta}_{(m+1)} \leftarrow \frac{1}{2}\tilde{\beta} + \frac{1}{2}\hat{\beta}_{(m)}$$
- Using this procedure, we can solve for  $\hat{\beta}$  in 6 iterations, using step-halving only once, on the initial update

## Guaranteed convergence?

- The procedure on the previous page almost always works, but is still not *guaranteed* to converge
- The reason is that step halving might not be enough: it is possible that  $\ell(\frac{1}{2}\tilde{\beta} + \frac{1}{2}\hat{\beta}_{(m)})$  is still smaller than  $\ell(\hat{\beta}_{(m)})$
- To guarantee convergence, we need to iteratively reapply the step-halving: consider  $\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \dots$  until we reach a step size small enough that the likelihood does, in fact, increase
- Typically, this is not necessary, but this kind of check is necessary to ensure that the likelihood goes up with every iteration, even in pathological cases

# Ties

- As a final comment, note that we are ignoring the issue of tied observations, even though there are in fact a few ties in the `pubc` data
- However, unless there are a large number of ties, this is typically a very minor issue:

	trt	stage	hepato	bili
Crude	-0.15530	0.62157	0.34860	0.13358
survival	-0.15473	0.62138	0.34854	0.13353

- We will, however, discuss ties more carefully in a future lecture