The Cox Proportional Hazards Model (CPH)

Leonid Shpaner[1]

[1]University of California, Los Angeles, lshpaner@ucla.edu

Abstract

The Cox Proportional Hazards (CPH) Model, also known as the Cox Model, is a statistical method used for survival analysis in medical research. It allows researchers to assess the effect of multiple factors on the time to an event of interest, such as death or disease onset, while controlling for other variables. The Cox Model is a type of regression model that estimates the hazard ratio, which is the ratio of the hazard rates between two groups. The hazard rate is the probability of experiencing an event of interest at a given time, given that the individual has survived up to that point in time.

Keywords: CPH Model, Survivability

1 Mathematical Formulation

The CPH model consists of a system of equations, which can be expressed mathematically as:

$$h(t|x) = h_o(t) \exp \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \tag{1}$$

$$h(t|x) = h_o(t) \exp\left((x - \bar{x})'\beta\right) \tag{2}$$

where $h(t|x)$ is the hazard hazard rate at time t for an individual with a set of predictor variables x, $h_o(t)$ is the baseline hazard rate at time t, which is the hazard rate when all predictor variables are set to 0. $\beta_1 x_1, \beta_2 x_2, \cdot \cdot \cdot \beta_p x_p$ are the coefficients for the predictor variables $x_1, x_2, x_3, \cdot \cdot \cdot, x_p$, and $\exp(\beta_1 x_1, \beta_2 x_2, \cdot \cdot \cdot \beta_p x_p)$ is the is the hazard ratio, which represents the proportional increase or decrease in the hazard rate for a one-unit increase in the predictor variable, holding all other variables constant.

2 Assumptions

The Cox Model assumes that the hazard ratio is constant over time, which means that the proportional effect of a predictor variable on the hazard rate remains the same over time.

The Cox Model estimates the values of $\beta_1 x_1, \beta_2 x_2, \cdot \cdot \cdot \beta_p x_p$ using maximum likelihood estimation and provides estimates of the hazard ratios and their confidence intervals. The proportional hazards assumption is tested by examining the Schoenfeld residuals, which should not show any significant pattern over time.

3 Schoenfeld Residuals

The Schoenfeld residuals are a type of residual that measure the difference between the observed covariate values and the expected covariate values for each individual at each event time.

The proportional hazards assumption states that the effect of a covariate on the hazard rate is constant over time. In other words, the hazard ratios for each covariate should remain constant over time. If this assumption is violated, it means that the effect of the covariate on the hazard rate is not proportional and may change over time.

To test the proportional hazards assumption using the Schoenfeld residuals, we can follow these steps:

1. Fit the CPH model using the Breslow method (or another method).
2. Calculate the Schoenfeld residuals for each covariate by regressing the residuals against the covariate values for each individual.
3. Plot the Schoenfeld residuals against time and examine whether there is a significant trend over time. If the proportional hazards assumption holds, the Schoenfeld residuals should not show any significant pattern over time.
4. Perform a formal test of the proportional hazards assumption by regressing the Schoenfeld residuals against time, and examining whether the coefficient for time is significantly different from zero. A significant coefficient indicates that the proportional hazards assumption is violated.

The regression of the Schoenfeld residuals against time can be written as:

$$r_{(it)} = \beta(t) z_i \tag{3}$$

where $r_{(it)}$ is the Schoenfeld residual for individual i at time t, $\beta(t)$ is a vector of regression coefficients that varies with time, and z_i is a vector of covariate values for individual i.

If the proportional hazards assumption holds, then $\beta_(t)$ should be equal to zero for all values of t. To test this, we can perform a Cox regression of the Schoenfeld residuals against time, and test whether the coefficient for time is significantly different from zero. This can be written as:

$$h(t|z) = h_0(t) \times \exp(\beta' z + \gamma t) \tag{4}$$

where $h(t|z)$ is the hazard rate at time t for a given set of covariate values z, $h_0(t)$ is the baseline hazard rate at time t , $\beta' z$ is the linear predictor term for the covariate values z, and γ is the coefficient for time, which represents the effect of time on the hazard rate.

If γ is significantly different from zero, it indicates that the effect of the covariate on the hazard rate is not proportional over time, and the proportional hazards assumption is violated.

To use the Cox Model, researchers need to have data on the time of the event of interest (such as the time of death or disease onset), as well as information on the predictor variables that may influence the hazard rate (such as age, sex, or treatment group). The model produces estimates of the hazard ratios for each predictor variable, along with confidence intervals and p-values to assess the statistical significance of the associations.

4 Breslow Method Using Python

The Breslow method, also known as the partial likelihood method, is a commonly used method for estimating the Cox proportional hazards model in survival analysis. The method is used to estimate the regression coefficients and corresponding hazard ratios for the predictor variables.

To demonstrate the Breslow method using Python, we can use the `lifelines` package, which is a popular Python library for survival analysis. Here is an example of how to use the `CoxPHFitter` class from `lifelines` to fit a CPH model using the Breslow method:

```
# Import necessary packages
import pandas as pd
from lifelines import CoxPHFitter

# Load survival data
data = pd.read_csv('survival_data.csv')

# Create a new instance of the CoxPHFitter class
cph = CoxPHFitter()

# Fit the Cox proportional hazards model using the Breslow method
cph.fit(data, 'time_to_event', event_col='event', show_progress=True)
```

In the above code, we first load our survival data into a Pandas DataFrame called `data`, which should include columns for the time to event (e.g. time to death or time to disease progression) and an event indicator variable (e.g. a binary variable indicating whether the event occurred). We then create a new instance of the `CoxPHFitter` class from the `lifelines` package and call the `fit` method to fit the CPH model using the Breslow method.

The `fit` method requires the following arguments:

`data`: the DataFrame containing the survival data.
`time_to_event`: the name of the column containing the time to event data.
`event_col`: the name of the column containing the event indicator data.

We can also specify other optional arguments, such as `show_progress=True` to display a progress bar during the fitting process. After fitting the model, we can access the estimated regression coefficients and corresponding hazard ratios using the summary method:

```
# Print summary of the fitted model
print(cph.summary)
```

This will display a table showing the estimated regression coefficients, hazard ratios, standard errors, and p-values for each predictor variable, along with other information about the model fit.

The Breslow method is the default `baseline_estimation_method` parameter in the CPH model, with `spline`, and `piecewise` being additional parameters.

5 Lasso Regularization

L1 regularization, also known as Lasso regularization, is a technique used in the CPH model to penalize the magnitude of the regression coefficients associated with the covariates. The *L1* penalty encourages some of the coefficients to be exactly zero, leading to a sparse model with fewer covariates. Mathematically, the CPH model with *L1* penalty can be expressed as:

$$L(\beta) = \prod_i \frac{h(t_i|x_i)}{\sum_j h(t_j|x_j)} \delta \exp(\beta_1 x_i + \beta_2 x_{2_i} + \cdots + \beta_p x_{pi}) \tag{5}$$

The *L1* penalty is added to the log-likelihood function of the CPH model as follows:

$$\log L(\beta') = \sum_i \delta_i \left(\beta' x_i - \log \sum_j \delta \exp(\beta' x_j) \right) - \lambda \sum_j |\beta_j| \tag{6}$$

where δ_i is an indicator variable that equals 1 if the i^{th} subject has an event and 0 otherwise and $|\beta_j|$ is the absolute value of the j^{th} coefficient The first term in the equation is the log-likelihood of the CPH model and the second term is the *L1* penalty which is a sum of the absolute values of the regression coefficients multiplied by the penalty parameter λ.

The objective of fitting the CPH model with *L1* penalty is to find the set of regression coefficients that maximizes the log-likelihood function subject to the c penalty. This is equivalent to minimizing the negative log-likelihood function plus the *L1* penalty, which can be solved using optimization algorithms such as coordinate descent or Least Angle Regression (LARS). The resulting set of coefficients can then be used to make predictions for new subjects.

In lifelines, the *L1*-penalized CPH model can be fit using the `CoxPHFitter` class with the penalizer parameter set to `'lasso'`. Here is an example of how to fit an *L1*-penalized CPH model in lifelines:

```python
from lifelines.datasets import load_rossi
from lifelines import CoxPHFitter

data = load_rossi()
cph = CoxPHFitter(penalizer=’lasso’, l1_ratio=1.0)
cph.fit(data, duration_col=’week’, event_col=’arrest’)
cph.print_summary()
```

In this example, we load the Rossi dataset, which contains information about 432 convicts released from Maryland state prisons who were followed up for up to 5 years after release. We then create an instance of the `CoxPHFitter` class and set the penalizer parameter to `'lasso'` and the `l1_ratio parameter` to 1.0 to perform *L1*-penalized CPH regression. Finally, we fit the model to the data using the `duration_col` and `event_col` arguments to specify the time-to-event and event indicator columns, and we print a summary of the model's coefficients and performance.

References

[1] Davidson-Pilon, C., & Lifelines contributors. (2021). *Lifelines: Survival analysis in Python*. Zenodo. https://doi.org/10.5281/zenodo.5562133