PH 716 Applied Survival Analysis

Part VIII: Model/variable/subset selection

Zhiyang Zhou (zhou67@uwm.edu, zhiyanggeezhou.github.io)

2024/Apr/06 20:39:41

Comparing nested models

- The value of (partial) log-likelihood always increasing with added covariates
  - But a parsimonious model preferred if the incremental (partial) log-likelihood is insignificant, because
    - Reduced overfitting
    - Increased computational efficiency
    - Enhanced stability (lower risk of multicollinearity)
    - Ease of application
    - Improved interpretability
- Comparing nested models $\Leftrightarrow$ testing the significance of the increment attributable to added covariates

Ex. 8.1: A toy example

- Dataset `asaur::pharmacoSmoking`
  - `ageGroup4`: Age group with levels `21-34`, `35-49`, `50-64`, or `65+`
  - `employment`: `ft` (full-time), `pt` (part-time), or `other`
- Candidate models
  - Model A: `ageGroup4`
  - Model B: `employment`
  - Model C: `ageGroup4+employment`
- Models A and B both nested to Model C

```
options(digits=4)
library(survival)
data.ex81 = asaur::pharmacoSmoking
fit.ex81.a = coxph(Surv(ttr, relapse)~ageGroup4, data=data.ex81)
fit.ex81.b = coxph(Surv(ttr, relapse)~employment, data=data.ex81)
fit.ex81.c = coxph(Surv(ttr, relapse)~ageGroup4+employment, data=data.ex81)
logLik(fit.ex81.a)
logLik(fit.ex81.b)
logLik(fit.ex81.c)

anova(fit.ex81.a, fit.ex81.c)
anova(fit.ex81.b, fit.ex81.c)
```

Comparing non-nested models

- Akaike Information Criterion (AIC)

$$\text{AIC} = -2p\ell(\hat{\beta}_1, \ldots, \hat{\beta}_p) + 2p$$

  - $p\ell(\hat{\beta}_1, \ldots, \hat{\beta}_k)$: (partial) log-likelihood evaluated at estimates $\hat{\beta}_1, \ldots, \hat{\beta}_k$
  - The smaller the better
- Bayesian Information Criterion (BIC)

$$\text{BIC} = -2p\ell(\hat{\beta}_1, \ldots, \hat{\beta}_p) + (\ln n)p$$

  - $p\ell(\hat{\beta}_1, \ldots, \hat{\beta}_k)$: (partial) log-likelihood evaluated at estimates $\hat{\beta}_1, \ldots, \hat{\beta}_k$
  - n: sample size
  - The smaller the better

Revisit Ex. 8.1

- Dataset `asaur::pharmacoSmoking`
  - `ageGroup4`: Age group with levels `21-34`, `35-49`, `50-64`, or `65+`
  - `employment`: `ft` (full-time), `pt` (part-time), or `other`
- Candidate models
  - Model A: `ageGroup4`
  - Model B: `employment`
  - Model C: `ageGroup4+employment`
- Model A not nested to B

```
AIC(fit.ex81.a)
AIC(fit.ex81.b)
AIC(fit.ex81.c)

BIC(fit.ex81.a)
BIC(fit.ex81.b)
BIC(fit.ex81.c)
```

Generic procedure of model selection

- Forward selection
  1. Start with the minimal model.
  2. Add predictors one by one: consider adding each of the remaining predictors and evaluate how much it improves the model. The evaluation metric can be the improvement in likelihood, AIC, BIC, or any other appropriate metric.
  3. Select the best predictor: Add the predictor that improves the model the most (e.g., results in the largest decrease in AIC).
  4. Repeat steps 2–3 until no significant improvement.
  5. Final model: The process stops when no adding improves the model.
- Backward selection (NOT always doable)
  1. Start with the maximal model.
  2. Remove predictors one by one: consider removing each predictor in the current model and evaluate how much it improves the model. The evaluation metric can be the improvement in likelihood, AIC, BIC, or any other appropriate metric.
  3. Select the best predictor: Delete the predictor that improves the model the most (e.g., results in the largest decrease in AIC).
  4. Repeat steps 2–3 until no significant improvement.
  5. Final model: The process stops when no deletion improves the model.
- (Bidirectional) stepwise selection
  1. Initiate by adding the best variable.
  2. After adding a new variable, it checks whether any of the previously included variables have become unnecessary and should be removed.
  3. Each forward selection step can be followed by one or more backward elimination steps.
  4. This process continues until no variable can be added/removed to improve the model.

Revisit Ex. 8.1

```
options(digits=4)
library(survival)
data.ex81 = asaur::pharmacoSmoking
minimal = coxph(Surv(ttr, relapse)~grp, data=data.ex81)
maximal = coxph(
  Surv(ttr, relapse)~grp+gender+race+employment+yearsSmoking+levelSmoking+ageGroup4,
  data=data.ex81
)
scope = list(lower = minimal, upper = maximal)
# Selection with AIC
forward.AIC = step(minimal, scope = scope, direction = "forward", k = 2)
backward.AIC = step(maximal, scope = scope, direction = "backward", k = 2)
stepwise.AIC = step(minimal, scope = scope, direction = "both", k = 2)
# Selection with BIC
forward.BIC = step(minimal, scope = scope, direction = "forward", k = log(nrow(data.ex81)))
backward.BIC = step(maximal, scope = scope, direction = "backward", k = log(nrow(data.ex81)))
stepwise.BIC = step(minimal, scope = scope, direction = "both", k = log(nrow(data.ex81)))
```

Penalization/regularization/shirinkage

- model selection $\Leftrightarrow$ variable selection $\Leftrightarrow$ compressing certain coefficients to zeros
- Introducing bias but may significantly decrease variances of the estimates
- Ridge: minimizing a penalized (partial) log-likelihood

$$-p\ell(\beta_1, \ldots, \beta_p) + \lambda \sum_{j=1}^{p} \beta_j^2$$

  with respect to $\beta_1, \ldots, \beta_p$

  - $\lambda \geq 0$: a tuning parameter
  - Penalizing large estimates of coefficients (BUT setting NO coefficient to exactly zero)
  - Equv. minimizing $-p\ell(\beta_1, \ldots, \beta_p)$ subject to $\sum_{j=1}^{p} \beta_j^2 \leq \eta(\lambda)$
    - $\eta > 0$: a scalar corresponding to λ

- Least absolute shrinkage and selection operator (lasso): minimizing a penalized (partial) log-likelihood

$$-p\ell(\beta_1, \ldots, \beta_p) + \lambda \sum_{j=1}^{p} |\beta_j|$$

  with respect to $\beta_1, \ldots, \beta_p$

  - $\lambda \geq 0$: a tuning parameter controlling the overall penalty strength
  - Shrinking some estimates of coefficients to zero
  - Equv. minimizing $-p\ell(\beta_1, \ldots, \beta_p)$ subject to $\sum_{j=1}^{p} |\beta_j| \leq \eta(\lambda)$
    - $\eta > 0$: a scalar corresponding to λ

- Elastic net: minimizing a penalized (partial) log-likelihood

$$-p\ell(\beta_1, \ldots, \beta_p) + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2 + \alpha \sum_{j=1}^{p} |\beta_j| \right)$$

  with respect to $\beta_1, \ldots, \beta_p$

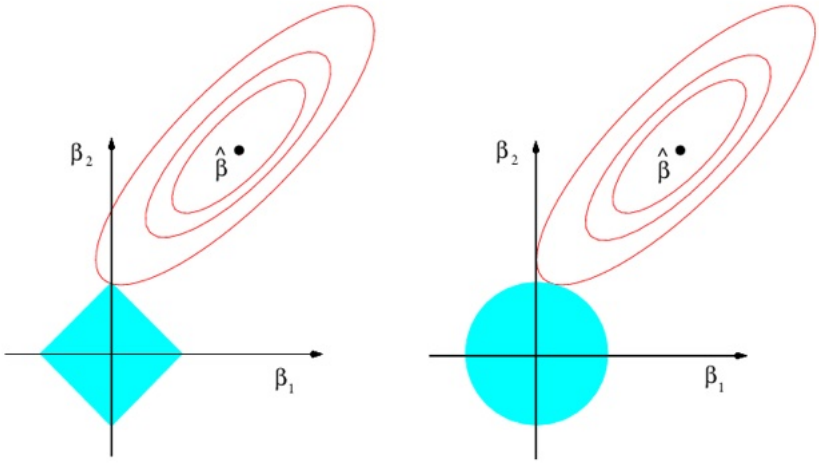

Figure 1: Contours of the error and constraint functions for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq \eta$ and $\beta_1^2 + \beta_2^2 \leq \eta$, while the red ellipses are the contours of (partial) log-likelihood. (G. James, D. Witten, T. Hastie, & R. Tibshirani, 2021, *An Introduction to Statistical Learning*, 2nd Ed., pp. 245)

  - $\lambda \geq 0$: a tuning parameter controlling the overall penalty strength
  - $\alpha \in [0, 1]$: a tuning parameter controlling the regression type
  - A mixture of ridge ($\alpha = 0$) and lasso ($\alpha = 1$)
- `R` function `glmnet::glmnet`

Revisit Ex. 8.1

```r
options(digits=4)
library(survival)
library(glmnet)
data.ex81 = asaur::pharmacoSmoking[
  complete.cases(asaur::pharmacoSmoking[
    ,
    c('grp','gender','race','employment','yearsSmoking','levelSmoking','ageGroup4')
  ]),
] # excluding rows with NAs
sapply(data.ex81, class)
maximal = coxph(
  Surv(ttr, relapse)~
    grp+gender+race+employment+yearsSmoking+levelSmoking+ageGroup4,
  data=data.ex81,
  x = T
)
zero.ttr.idx = (data.ex81$ttr == 0 & data.ex81$relapse == 1) # subjects with event time 0

# ridge-penalized Cox PH (with the lambda sequence generated automatically)
cox.ridge = glmnet(
  x=maximal$x[!zero.ttr.idx,],
  y=maximal$y[!zero.ttr.idx],
  family = 'cox',
  alpha = 0, # ridge
```

```r
  nlambda = 100 # number of different lambda values
)
## paths of estimates against log(lambda)
plot(cox.ridge, xvar="lambda", label=TRUE)

# lasso-penalized Cox PH (with the lambda sequence generated automatically)
cox.lasso = glmnet(
  x=maximal$x[!zero.ttr.idx,],
  y=maximal$y[!zero.ttr.idx],
  family = 'cox',
  alpha = 1, # lasso
  nlambda = 100 # number of different lambda values
)
## paths of estimates against log(lambda)
plot(cox.lasso, xvar="lambda", label=TRUE)
```

Cross-validation (CV)

- To determine the value of tuning parameter(s)
- K-fold CV
  1. Partition the dataset into K subsets (called "folds")
  2. Set up a grid of values of λ
  3. For each potential value of λ
     a. Take one fold as a test set and the remaining $K-1$ folds as a training set.
     b. Fit the model on the training set.
     c. Evaluate the model fitting in terms of certain measure on the test set.
     d. Repeat a–c for all K folds and compute the average measure value, say $\text{CV}(\lambda) = n^{-1}\sum_{k=1}^{K}\text{measure}_k$.
  4. Select the value of λ that minimizes $\text{CV}(\lambda)$.
- Leave-one-out CV $\Leftrightarrow$ n-fold CV

Measures evaluating Cox PH models

- (Partial) likelihood deviance: $2\{p\ell_{\text{saturated}} - p\ell(\beta_1, \ldots, \beta_p)\}$
  - $p\ell_{\text{saturated}} = -\sum_m d_m \ln d_m$ with d_m as the number of events at the mth ordered failure time
- Concordance index (C-index)

$$Cidx = \frac{\sum_{i_1,i_2} \delta_{i_2} \cdot 1_{\tilde{t}_{i_1} > \tilde{t}_{t_2}} \cdot 1_{\lambda_{i_1} < \lambda_{t_2}}}{\sum_{i_1,i_2} \delta_{i_2} \cdot 1_{\tilde{t}_{i_1} > \tilde{t}_{t_2}}}$$

  - Where
    - $1_{\tilde{t}_{i_1} > \tilde{t}_{t_2}} = 1$ if $\tilde{t}_{i_1} > \tilde{t}_{t_2}$ and 0 otherwise
    - $1_{\lambda_{i_1} < \lambda_{t_2}}$ if $\lambda_{i_1} < \lambda_{t_2}$ and 0 otherwise
  - To evaluate the discrimination/concordance of a survival model
    - Discrimination: the ability to correctly provide a reasonable ranking of survival times based on the predicted individual risks
    - Concordance: the subject with a higher predicted risk corresponds to a longer lifetime
  - Ranging from .5 to 1, the higher the better
    - $Cidx = .5$: a model with no predictive ability, equivalent to random guessing
    - $Cidx = 1$: a perfect model where the predictions and outcomes are in complete concordance
- Time-dependent Receiver Operating Characteristic (ROC) curve
  - Calculation: for a given time point, say t

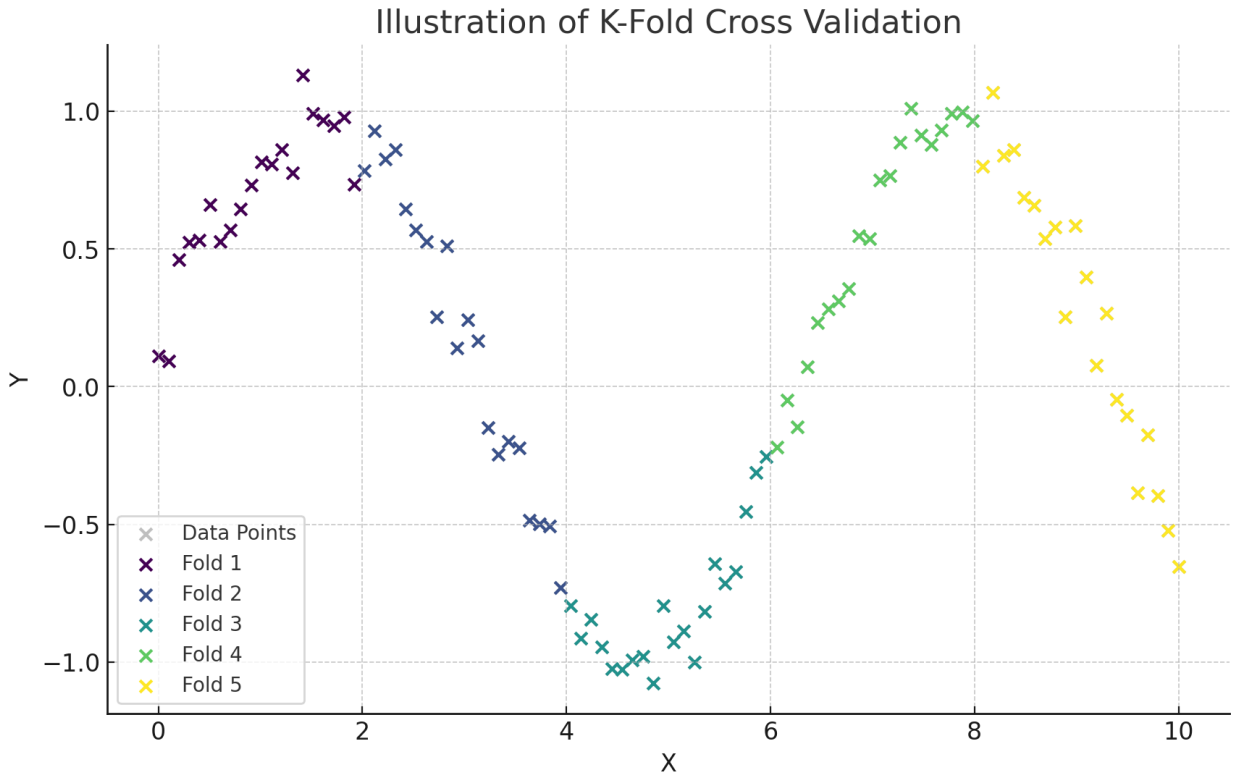

Figure 2: 5-fold CV

1. Sorting individuals by their risk scores
2. Iteratively using each unique risk score as a threshold
3. Calculating sensitivity and specificity at each threshold.

   * Sensitivity (true positive rate; the model's ability to correctly identify individuals who experience the event): the proportion of individuals who experienced the event before time t and whose risk score exceeds the threshold
   * Specificity (true negative rate; the model's ability to correctly identify individuals who do not experience the event): the proportion of individuals who have not experienced the event by time t (including those censored before t) and whose risk score is below the threshold

  – A generalization of ROC for binary classification

- Time-dependent Area Under the Curve (AUC)
  – The area under the time-dependent ROC curve
  – Alternative to C-index in evaluating the discrimination ability
  – A generalization of AUC for binary classification
- `R` function `glmnet::cv.glmnet`

Revisit Ex. 8.1

```
options(digits=4)
library(survival)
library(glmnet)
data.ex81 = asaur::pharmacoSmoking
sapply(data.ex81, class)
maximal = coxph(
  Surv(ttr, relapse)~
```

```
    grp+gender+race+employment+yearsSmoking+levelSmoking+ageGroup4,
  data=data.ex81,
  x = T
)
concordance(maximal)$concordance # C-index
roc = timeROC::timeROC(
  T = data.ex81$ttr,
  delta = data.ex81$relapse,
  marker = predict(maximal, newdata = data.ex81, type = 'lp'), # risk scores
  times = 1:100, # time range of interest
  cause = 1 # the label of event
)
plot(roc, time=10) # time-dependent ROC curve at time 10
roc$AUC['t=10'] # time-dependent AUC at time 10
plot( # plot of time-dependent AUCs at times 1:100
  x=1:100, y=roc$AUC,
  xlab = 'Time', ylab = 'Time-dependent AUC',
  type = 'l'
)

# 5-fold CV
zero.ttr.idx = (data.ex81$ttr == 0 & data.ex81$relapse == 1) # subjects with zero event time
cv.fit = cv.glmnet(
  x=maximal$x[!zero.ttr.idx,],
  y=maximal$y[!zero.ttr.idx],
  family = 'cox',
  alpha = 1, # lasso
  type.measure = c("deviance"), # "deviance" or "C"
  nfolds = 5
)
cv.fit$lambda.min
cv.fit$lambda # values of lambda used in the fits
cv.fit$cvm # mean cross-validated errors
plot(cv.fit)

# fit with the lambda given by CV
cox.lasso = glmnet(
  x=maximal$x[!zero.ttr.idx,],
  y=maximal$y[!zero.ttr.idx],
  family = 'cox',
  alpha = 1,
  lambda = cv.fit$lambda.min
)
cox.lasso$beta

# C-index of the model fitted via glmnet
pred.risk.scores <- predict(
  cox.lasso, newx = maximal$x[!zero.ttr.idx,],
  type = "link",
  s = cv.fit$lambda.min
)
Cindex(pred = pred.risk.scores, y = maximal$y[!zero.ttr.idx])
```

Trade-off between prediction accuracy and model interpretability

- When the goal is understanding how predictors are associated with outcomes: simpler, inflexible models preferred due to their interpretability
- When the goal is solely prediction: flexible models potentially becoming the best choice
  - BUT more flexible models not always leading to more accurate predictions; highly flexible models susceptible to overfitting, potentially capturing noise in the training data as if it were a real pattern