Module 4: Semi and Weakly Supervised Learning

Jianhui Gao, Siyue Yang, and Jessica Gronsbell

01/06/2022

```r
# If a package is installed, it will be loaded. If any
## are not, the missing package(s) will be installed
## from CRAN and then loaded.

## First specify the packages of interest
packages <- c(
  "dplyr", "PheCAP", "glmnet", "randomForestSRC", "PheNorm",
  "MAP", "pROC", "mltools", "data.table", "ggplot2", "parallel"
)

## Now load or install&load all
package.check <- lapply(
  packages,
  FUN = function(x) {
    if (!require(x, character.only = TRUE)) {
      install.packages(x, dependencies = TRUE)
      library(x, character.only = TRUE)
    }
  }
)

# load environment from example 1
load("../module4/environment_pass.RData")
source("../Rscripts/helper_function.R")
```

Semi-supervised Learning

(i) Regress the surrogate on the features with penalized least square to get the direction of beta.

```r
x <- all_x %>% select(starts_with("COD") | starts_with("NLP")) # COD + NLP
S <- ehr_data$surrogate

# Step 1
beta.step1 <- adaptive_lasso_fit(
  y = S, # surrogate
  x = x, # all X
  family = "gaussian",
  tuning = "cv"
)

# Features selected
names(beta.step1[abs(beta.step1) > 0])[-1]
```

```
##   [1] "COD2"   "COD10"  "NLP2"   "NLP3"   "NLP5"   "NLP7"   "NLP15"  "NLP21" 
##   [9] "NLP24"  "NLP28"  "NLP29"  "NLP47"  "NLP51"  "NLP56"  "NLP61"  "NLP68" 
##  [17] "NLP74"  "NLP78"  "NLP81"  "NLP93"  "NLP94"  "NLP95"  "NLP104" "NLP127"
##  [25] "NLP140" "NLP146" "NLP150" "NLP160" "NLP161" "NLP164" "NLP172" "NLP176"
##  [33] "NLP178" "NLP179" "NLP183" "NLP185" "NLP186" "NLP192" "NLP195" "NLP200"
##  [41] "NLP202" "NLP211" "NLP212" "NLP215" "NLP220" "NLP225" "NLP231" "NLP234"
##  [49] "NLP243" "NLP250" "NLP252" "NLP266" "NLP281" "NLP287" "NLP291" "NLP292"
##  [57] "NLP294" "NLP297" "NLP299" "NLP301" "NLP302" "NLP304" "NLP306" "NLP309"
##  [65] "NLP318" "NLP321" "NLP325" "NLP326" "NLP334" "NLP338" "NLP349" "NLP350"
##  [73] "NLP357" "NLP359" "NLP361" "NLP367" "NLP380" "NLP387" "NLP395" "NLP398"
##  [81] "NLP403" "NLP405" "NLP430" "NLP431" "NLP434" "NLP437" "NLP438" "NLP446"
##  [89] "NLP451" "NLP456" "NLP457" "NLP463" "NLP465" "NLP466" "NLP467" "NLP470"
##  [97] "NLP473" "NLP482" "NLP483" "NLP484" "NLP486" "NLP490" "NLP495" "NLP516"
## [105] "NLP519" "NLP520" "NLP523" "NLP529" "NLP533" "NLP541" "NLP544" "NLP547"
## [113] "NLP556" "NLP560" "NLP561" "NLP562" "NLP564" "NLP574"
```

(ii) Regress the outcome on the linear predictor to get the intercept and multiplier for the beta.

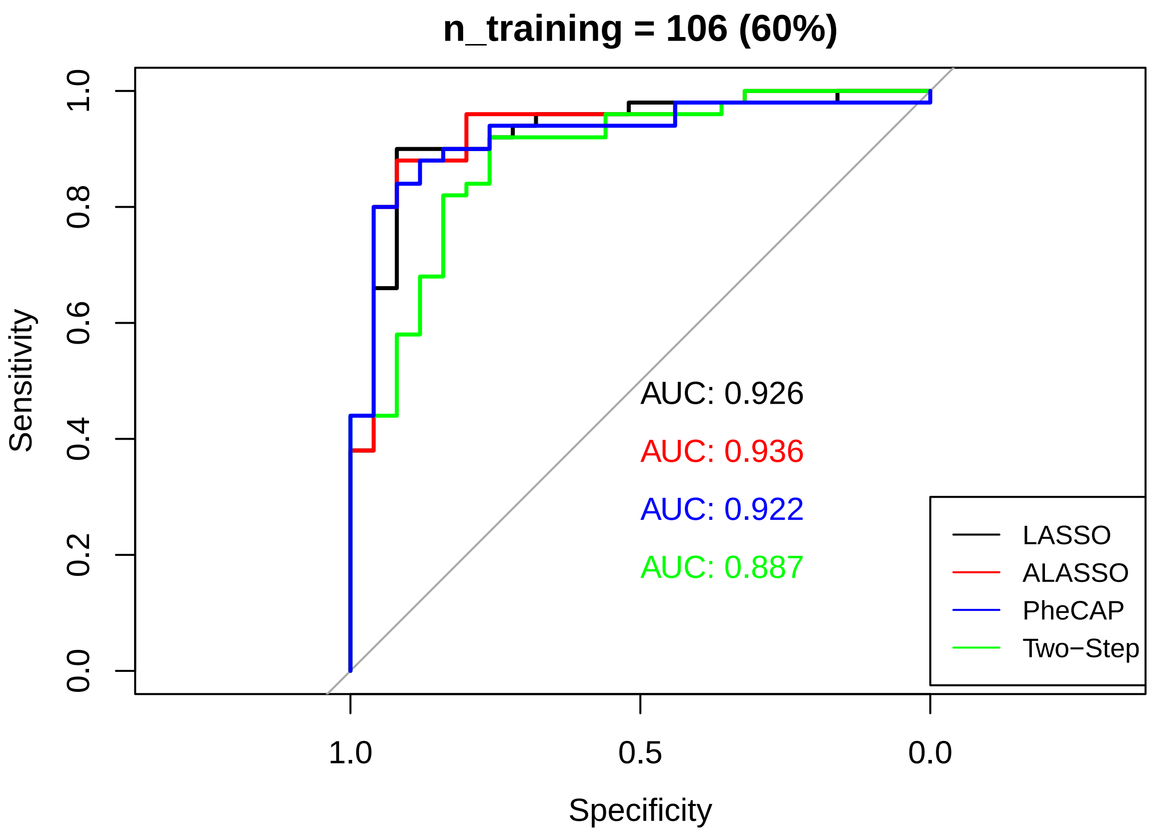
```
# linear predictor without intercept
bhatx <- linear_model_predict(beta = beta.step1, x = as.matrix(x))

# Step 2
step2 <- glm(train_y ~ bhatx[train_data$patient_id] + S[train_data$patient_id] +
  health_count[train_data$patient_id])
beta_step2 <- coef(step2)
beta_step2
```

```
##                          (Intercept)        bhatx[train_data$patient_id]
##                           1.00617125                          0.09555835
##             S[train_data$patient_id] health_count[train_data$patient_id]
##                           0.14529744                         -0.11451665
```

```
# recover beta
beta <- beta_step2[2] * beta.step1
# mu
mu <- beta_step2[1] +
  as.numeric(as.matrix(x[test_data$patient_id, ]) %*% beta[-1]) +
  as.numeric(beta_step2[3] %*% S[test_data$patient_id]) +
  as.numeric(beta_step2[4] %*% health_count[test_data$patient_id])
# expit
y_hat.ss <- plogis(mu)
```

```
plot(roc(test_y, y_hat.lasso),
  print.auc = TRUE, main = "n_training = 106 (60%)"
)
plot(roc(test_y, y_hat.alasso),
  print.auc = TRUE, col = "red", add = TRUE, print.auc.y = 0.4
)
plot(roc(test_y, y_hat.ss),
  print.auc = TRUE, col = "green", add = TRUE, print.auc.y = 0.2
)
plot(roc(test_y, y_hat.phecap),
  print.auc = TRUE, col = "blue", add = TRUE, print.auc.y = 0.3
)
legend(0, 0.3,
  legend = c("LASSO", "ALASSO", "PheCAP", "Two-Step"),
  col = c("black", "red", "blue", "green"),
  lty = 1, cex = 0.8
)
```



```
ss.roc.full <- get_roc(test_y, y_hat.ss)
head(ss.roc.full, 10)
```

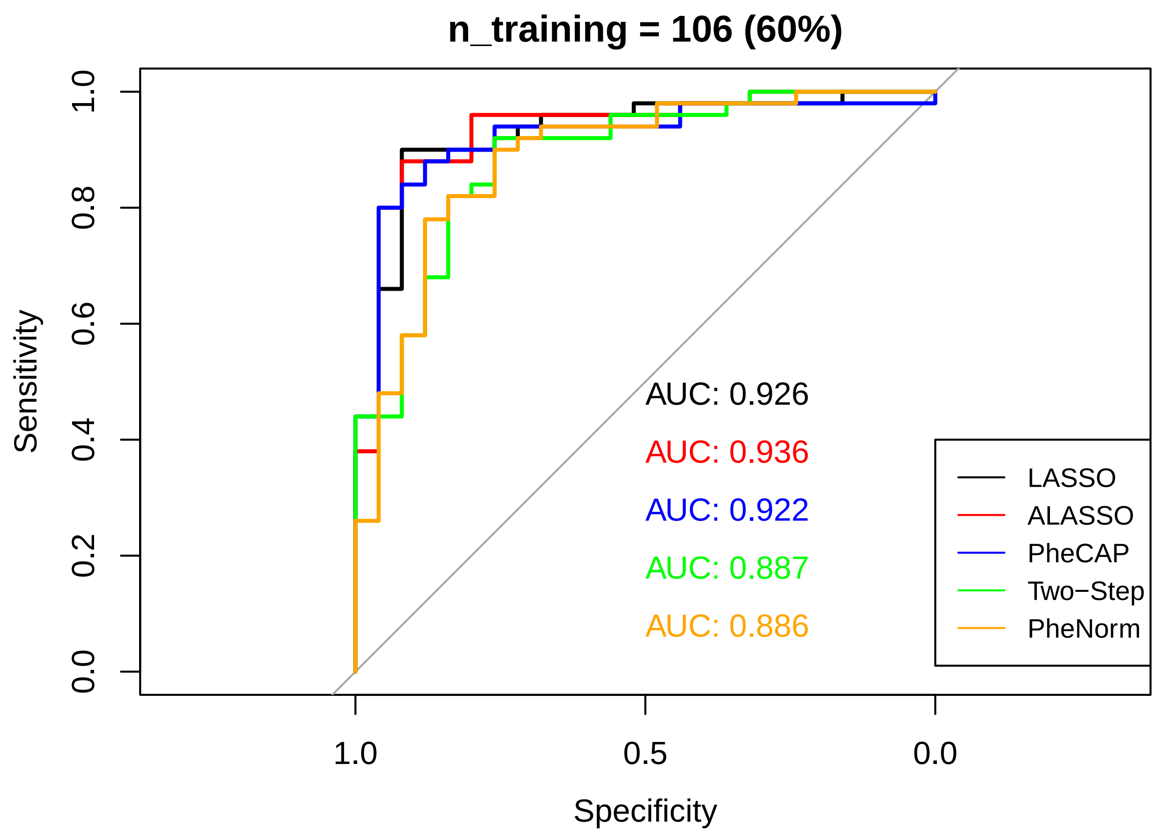
```
##          cutoff    pos.rate  FPR       TPR       PPV       NPV        F1
##  [1,] 0.7759103 0.006666667 0.00 0.2104348 1.0000000 0.3877276 0.3477011
##  [2,] 0.7408614 0.093333333 0.00 0.3252174 1.0000000 0.4256107 0.4908136
##  [3,] 0.7058124 0.300000000 0.02 0.4400000 0.9777778 0.4666667 0.6068966
##  [4,] 0.7052806 0.306666667 0.04 0.4400000 0.9565217 0.4615385 0.6027397
##  [5,] 0.7047487 0.306666667 0.04 0.4400000 0.9565217 0.4615385 0.6027397
##  [6,] 0.7042169 0.306666667 0.04 0.4400000 0.9565217 0.4615385 0.6027397
##  [7,] 0.7036850 0.313333333 0.06 0.4400000 0.9361702 0.4563107 0.5986395
##  [8,] 0.7033164 0.320000000 0.08 0.4750000 0.9223301 0.4670051 0.6270627
##  [9,] 0.7029477 0.326666667 0.08 0.5100000 0.9272727 0.4842105 0.6580645
## [10,] 0.6994688 0.386666667 0.08 0.5450000 0.9316239 0.5027322 0.6876972
```

Weakly supervised learning

```
model_phenorm <- PheNorm.Prob(
  nm.logS.ori = "surrogate", # name of surrogates
  nm.utl = "healthcare_utilization", # name of HU
  nm.X = colnames(ehr_data)[-1:-4], # Other predictors X
  dat = ehr_data,
  train.size = nrow(ehr_data)
)
```

```
y_hat.phenorm <- model_phenorm$probs[data$validation_set]
plot(roc(test_y, y_hat.lasso),
  print.auc = TRUE, main = "n_training = 106 (60%)"
)
plot(roc(test_y, y_hat.alasso),
  print.auc = TRUE, col = "red", add = TRUE, print.auc.y = 0.4
)
plot(roc(test_y, y_hat.phecap),
  print.auc = TRUE, col = "blue", add = TRUE, print.auc.y = 0.3
)
plot(roc(test_y, y_hat.ss),
  print.auc = TRUE, col = "green", add = TRUE, print.auc.y = 0.2
)
plot(roc(test_y, y_hat.phenorm),
  print.auc = TRUE, col = "orange", add = TRUE, print.auc.y = 0.1
)

legend(0, 0.4,
  legend = c("LASSO", "ALASSO", "PheCAP", "Two-Step", "PheNorm"),
  col = c("black", "red", "blue", "green", "orange"),
  lty = 1, cex = 0.8
)
```



Can not run MAP, MAP uses poisson regression. Requires integer count data.