Pairwise-Fused-Lasso

2025-05-21

Formulation

Question: built a regression model with sparsity for each cancer type, and group genes in the same cluster together. Proposed Solution:

For each cancer type

1. Let y_i be a binary variable representing the if subject i has cancer or not. $y_i = 1$ if unit i has the target cancer, 0 otherwise.
2. Let $x_{i,j}$ be a continuous variable representing the DNA methylation percentage at gene j in subject i
3. Let β_j be the regression coefficient for gene j
4. $y_i|x_i \sim Bernoulli(g_\beta(x_i))$, where $g_\beta(x_i) = logit(-x_i^T\beta) = \frac{1}{1+\exp(-x_i^T\beta)}$.

Since the outcome variable is binary, we can formulate the optimization problem as a logistic regression with L1 penalty and a total-variation penalty:

$$\min_{\beta \in \mathbb{R}^p}\{-l(\beta; X) + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{1 \le j,k \le p, j \neq k} |\beta_j - \beta_k|\}$$

where $l(\beta; X)$ is the log-likelihood of β in the logistic regression model:

$$l(\beta; X) = \sum_{i=1}^{N}[y_i \cdot (x_i^T\beta) - log(1 + \exp(x_i^T\beta))]$$

and $\lambda_1 > 0, \lambda_2 > 0$ are turning parameters.

Complication:

The formulation looks similar to fused lasso, but not exactly because fused lasso assumes that $\{\beta_j\}$ has the natural ordering, but genes don't have ordering. We cannot applied fused lasso directly. Additionally, since would like to estimate underlying genetic graphs that contribute to the cancer incidence, glmnet package can only handle the lasso and elacstic net, but not fused lasso or graph guided version.

Remedy: Use CVXR package to solve optimization problem with custom loss function.

Attempt

```r
seCount <-  readRDS(file = "Common_pan_cancer_hyper_bins_adjusted_and_normalized_cnt_in_SE.RDS")
dim(seCount)
```

```
## [1] 24418   378
```

```r
sampleInfo <- read.csv("Common_pan_cancer_hyper_bins_adjusted_cnt_in_SE_samples_info.csv")
cancerTypes <- sampleInfo$cancer_type[sampleInfo$sample_id %in% colnames(seCount)]


## Load gene numbers that are important in PCA
load("impGenes.RData")
X <- t(seCount[impGenes, ])

## First try predicting any cancer, regardless of types
y <- ifelse(cancerTypes != "Normal", 1, 0)

library(CVXR)
```

```
## Warning: package 'CVXR' was built under R version 4.4.3

##
## Attaching package: 'CVXR'

## The following object is masked from 'package:stats':
##
##     power
```

```r
## Decision Variables
p <- length(impGenes)
beta <- Variable(p)

## Logistic Loss Function
negLogLklyhood <- -(
  sum(X[y == 1, ] %*% beta) +
    sum(logistic(-X %*% beta))
  )

## Penalty Function
pairwise_penalty <- function(beta, lambda_1 = 1, lambda_2 = 1) {
  L1 <- p_norm(beta, 1) * lambda_1

  p <- beta@dim[1]
  pair_indices <- combn(p, 2, simplify = FALSE)
  browser()
  # Create penalty matrix: each row enforces beta_j - beta_k
  penalty_matrix <- do.call(rbind, lapply(pair_indices, function(idx) {
    vec <- rep(0, p)
    vec[idx[1]] <- 1
    vec[idx[2]] <- -1
    vec
  }))
  fusionPenalty <- sum(abs(penalty_matrix %*% beta)) * lambda_2

  # fusionPenalty <- do.call(sum, lapply(pair_indices, function(idx) {
  #   abs(beta[pair_indices[1]] - beta[pair_indices[2]])
  # }))
  # fusionPenalty <- fusionPenalty * lambda_2
  L1 + fusionPenalty
}

## Problem Formulation
lambda_1 <- 1
lambda_2 <- 1
obj <- negLogLklyhood + pairwise_penalty(beta, lambda_1, lambda_2)
```

```
## Called from: pairwise_penalty(beta, lambda_1, lambda_2)
## debug: penalty_matrix <- do.call(rbind, lapply(pair_indices, function(idx) {
##     vec <- rep(0, p)
##     vec[idx[1]] <- 1
##     vec[idx[2]] <- -1
##     vec
## }))
## debug: fusionPenalty <- sum(abs(penalty_matrix %*% beta)) * lambda_2
## debug: L1 + fusionPenalty
```

```r
prob <- Problem(Minimize(obj))

## Solving
try({
  result <- solve(prob)
  ## Inspect Results
  betaEst <- result$getValue(beta)
  plot(1:p, sort(betaEst))
})
```

```
## Error in construct_intermediate_chain(object, candidate_solvers, gp = gp) :
##   Problem does not follow DCP rules.
```

The issue: although penalty terms are convex, but negative log likelihood is not convex, violating convex assumption in CVXR.