ST509_Midterm_HJ

Hwijun Kwon

2024-04-20

1. Introduction

$$y_i|x_i \sim Poisson(\mu_\beta(x_i))$$

test

2. Competing Methods

Poisson Regression

1. Unpenalized Poisson Regression

```
library(MASS)
generate_data <- function(n, p, nu) {
  mu <- rep(0, p)
  sigma <- outer(1:p, 1:p, FUN = function(i, j) nu^(abs(i-j)))
  x <- mvrnorm(n=n, mu=mu, Sigma=sigma)
  beta0 = 1
  beta = c(rep(1,3), rep(0, p-3))
  mu_x <- exp(beta0 + x %*% beta)
  y <- rpois(n, mu_x)
  return(list(x = x, y = y))
}
set.seed(2024020409)
train_data <- generate_data(500, 50, 0.7)
test_data <- generate_data(500, 50, 0.7)
```

```
my_poi <- function(X, y, init = NULL, max_iter = 1000, eps = 1e-5) {
  # Scaling
  y_sd <- sd(y); X_sd <- apply(X, 2, sd)
  y <- y /y_sd; X <- t(t(X)/ X_sd)
  # Add intercept term
  X <- cbind(1, X) ; n <- nrow(X); p <- ncol(X)
  # INIT
  if (is.null(init)) init <- rep(0, p);
  beta <- init
  # Iteration
  for (iter in 1:max_iter) {
    eta <- X %*% beta
    w <- exp(eta) # Mean = Variance

    # Limit w
```

```
    w <- pmin(w, 1e6)
    w <- pmax(w, 1e-6)

    z <- eta + (y-w)/w
    # IRLS
    X_tilde <- diag(c(sqrt(w))) %*% X
    z_tilde <- diag(c(sqrt(w))) %*% z
    qr_obj <-  qr(X_tilde)
    new_beta <- backsolve(qr_obj$qr, qr.qty(qr_obj, z_tilde)) # Check Convergence
  if (max(abs(new_beta - beta))/ max(abs(beta)) < eps) break
    beta <- new_beta
  }
  # Warning
  if (iter == max_iter) warning("Algorithm may not have converged!")
  # Restore beta coef
  beta <- c(beta) * c(1, 1/X_sd) + c(log(y_sd), rep(0, p-1))
  # Result
  list(X_tilde = dim(X_tilde), est = t(beta), iterations = iter)
}
```

```
train_data <- generate_data(500, 50, 0.7)
y <- train_data$y ; x <-train_data$x
my_poi(x, y, init = rep(3, 51))
```

```
## $X_tilde
## [1] 500  51
##
## $est
##           [,1]     [,2]      [,3]     [,4]        [,5]       [,6]        [,7]
## [1,] 0.9825567 1.003834 0.9952995 1.014792 -0.01483829 0.02260662 -0.03286948
##            [,8]         [,9]      [,10]        [,11]       [,12]      [,13]
## [1,] 0.01534072 -0.009470657 0.01184592 4.996025e-05 -0.03623518 0.03685734
##             [,14]       [,15]         [,16]       [,17]       [,18]      [,19]
## [1,] -0.008090208 0.008525065 -0.0003296124 0.006840637 -0.03829035 0.04146782
##            [,20]        [,21]       [,22]        [,23]       [,24]       [,25]
## [1,] -0.01138962 -0.005552458 0.001087698 0.0008345985 0.004879776 0.004859562
##            [,26]      [,27]       [,28]       [,29]        [,30]      [,31]
## [1,] -0.02066295 0.03018091 -0.01809376 0.002426706 -0.006006737 -0.0090391
##           [,32]       [,33]      [,34]       [,35]       [,36]      [,37]
## [1,] 0.00201061 -0.01714482 0.02848699 -0.01523338 -0.01929848 0.02232584
##            [,38]       [,39]      [,40]      [,41]       [,42]       [,43]
## [1,] 0.008763946 0.002053838 0.01180394 0.01938962 -0.01431178 -0.01536064
##           [,44]       [,45]        [,46]        [,47]        [,48]        [,49]
## [1,] 0.02951308 -0.03035566 0.0002294486 -0.008802028 -0.007037254 -0.008278157
##           [,50]        [,51]
## [1,] 0.01793886 -0.007693518
##
## $iterations
## [1] 161
```

```
for (i in seq(0.1, 3, 0.5)){

  # Define Initial Beta
  init = rep(i, 51)
```

```
  # my_poi
  result = my_poi(x, y, init = init)

  # print
  print(i)
  #print(result)

}
```

```
## [1] 0.1
## [1] 0.6
## [1] 1.1
## [1] 1.6
## [1] 2.1
## [1] 2.6
```