Poisson Regresion and its penalization

2023-04-30

데이터는 포아송회귀 가정에 의해 $\log\{\mu_{\boldsymbol{\beta}}(\boldsymbol{x}_i)\} = \beta_0 + \boldsymbol{\beta}^T\boldsymbol{x}_i$에 대하여

$$y_i|\boldsymbol{x}_i \sim \text{Poisson}(\mu_{\boldsymbol{\beta}}(\boldsymbol{x}_i)), \quad i = 1, \cdots, n$$

를 따른다고 하자.

독립변수는 $\boldsymbol{x}_i \sim^{iid} N_p(\mathbf{0}, \Sigma)$ 를 따르며, 여기서 $\{\Sigma\}_{ij} = \sigma_{ij} = \rho^{|i-j|}$이며 실험에서는 $\rho \in \{0, 0.7\}$로 지정하였다. 각 회귀계수는 $\beta_0 = 1$ 그리고 $\boldsymbol{\beta} = (1, 1, 1, 0, \cdots, 0)^T \in \mathbb{R}^p, p = 3, 50$으로 둔다. 훈련데이터는 $n = 500$, 시험데이터는 $n_{ts} = 5000$으로 설정하며, 포아송 회귀모형과 RIDGE, LASSO, elastic-net, SCAD 벌점항이 있는 포아송 회귀모형을 비교한다. 벌점화 모형에서 최적의 파라미터는 시험데이터의 deviance를 가장 낮게하는 파라미터로 정하였다. 전 과정을 100번 반복하여 정확도와 변수선택의 성능을 측정한다.

1. 데이터 생성

```r
library(mvtnorm) # 다변량 정규분포 난수생성 함수를 위한 패키지

sim_data = function(train_N = 500, test_N = 5000, p, rho){
  N = train_N + test_N
  if(p == 3){
    beta = rep(1, 3)
    cov = matrix(0, p, p)
    for(i in 1:p){
      for(j in 1:p){
        cov[i,j] = rho^{abs(i-j)}
      }
    }
    X = rmvnorm(N, mean = rep(0,p), sigma = cov)
  }

  if(p == 50){
    beta = c(rep(1, 3), rep(0, p-3))
    cov = matrix(0, p, p)
    for(i in 1:p){
      for(j in 1:p){
        cov[i,j] = rho^{abs(i-j)}
      }
    }
    X = rmvnorm(N, mean = rep(0,p), sigma = cov)
  }

  mu = exp(cbind(1,scale(X)) %*% c(1,beta))
  y = rpois(nrow(X), mu)

  train_id = sample(1:N, train_N)
  train_X = X[train_id,]
```

```
  test_X = X[-train_id,]

  train_y = y[train_id]
  test_y = y[-train_id]

  train_mu = mu[train_id,]
  test_mu = mu[-train_id,]

  return(list(train_X = train_X, train_y = train_y, train_mu = train_mu,
              test_X = test_X, test_y = test_y, test_mu = test_mu, beta = beta))
}

# 데이터 생성
sim_dat = sim_data(train_N = 500, test_N = 5000, p = 3, rho = 0)  # 데이터 생성
true_beta = c(1, sim_dat$beta)

tr_X = sim_dat$train_X
tr_y = sim_dat$train_y
te_X = sim_dat$test_X
te_y = sim_dat$test_y
```

2. 포아송 회귀모형

포아송 회귀모형의 로그-가능도함수는 다음과 같다:

$$
\begin{aligned}
\ell(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \log f(y_i; \mu_i) \\
&= \sum_{i=1}^{n} [-\mu_i + y_i \log \mu_i - \log y_i!] \\
&= \sum_{i=1}^{n} [-\exp(\boldsymbol{x}_i^T \boldsymbol{\beta}) + y_i \boldsymbol{x}_i^T \boldsymbol{\beta} - \log y_i!]
\end{aligned}
$$

뉴튼-랩슨 방법을 사용하기 위해 로그-가능도함수의 회귀계수에 대한 1차 도함수와 2차 도함수를 구하면 다음과 같다:

$$
\begin{aligned}
\frac{\partial \ell}{\partial \beta_j} &= \sum_{i=1}^{n} x_{ij}(y_i - \mu_i) \\
\frac{\partial \ell^2}{\partial \beta_j \partial \beta_k} &= -\sum_{i=1}^{n} x_{ij} x_{ik} \mu_i, \quad j, k = 1, \cdots, p
\end{aligned}
$$

그러면 위의 값들이 gradient $\nabla \ell(\boldsymbol{\beta}^{(t)})$와 Hessian $H^{-1}(\boldsymbol{\beta}^{(t)})$의 원소가 되어 다음 과정을 수렴할때까지 반복하여 추정치를 구할 수 있다:

$$
\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - H^{-1}(\boldsymbol{\beta}^{(t)}) \nabla \ell(\boldsymbol{\beta}^{(t)})
$$

```
fit_poisson_resgression = function(X, y){
  p = ncol(X)
  sX = scale(X)
  # initialize
  bet = runif(p+1)
  desX = cbind(1, sX)

  t = loss = 0
```

```r
  while(t < 200){
    t = t + 1
    mu = exp(desX %*% bet)

    # 1. 일차도함수 계산
    # dlogL = rep(0, p+1)
    # for(i in 1:(p+1)) dlogL[i] = sum(desX[,i] %*% (y - mu))
    dlogL = sapply(1:(p+1), function(i) sum(desX[,i] %*% (y - mu)))
    # 2. 이차도함수 계산
    H = matrix(0, p+1, p+1)
    for(i in 1:(p+1)){
      for(j in 1:(p+1)){
        H[i,j] = -sum(mu * (desX[,i] * desX[,j]))
      }
    }
    if(sum(H == -Inf) >= 1 | sum(is.nan(H)) >= 1) break

    bet_new = bet - ginv(H) %*% dlogL

    loss_new = sum(dpois(y, lambda = mu, log = T))

    if(loss_new == Inf | loss_new == -Inf) break
    if(max(abs(bet_new - bet))/max(abs(bet)) < 1e-7) break
    bet = bet_new
  } # end while

  return(list(X = X, y = y, beta = as.vector(bet), iter=t))
}

# glm 함수와의 비교
fit_poisson_resgression(tr_X, tr_y)$beta
```

```
## [1] 0.9693270 1.0040349 0.9937588 1.0219503
```

```r
glm(tr_y ~ tr_X, family = "poisson")$coefficient
```

```
## (Intercept)       tr_X1       tr_X2       tr_X3
##   0.9993287   0.9720346   1.0009003   1.0064354
```

3. 벌점화 포아송 회귀모형

3-1. elastic-net 벌점항

포아송 확률변수 Y_i의 평균과 분산은 각각 $\mathbb{E}(Y_i) = \mu_i$, $V(Y_i) = \mu_i$이다. 그리고 GLM 구조에서 분산 함수는 $V(\mu_i) = \mu_i$이고, 연결함수는 $\eta_i : (0, \infty) \mapsto (-\infty, \infty)$이며 결론적으로 다음과 같은 형태를 갖는다:

$$\eta_i = \log \mu_i$$

포아송 회귀에서의 연결함수 η_i, working response z_i와 weights w_i는 다음과 같이 정리된다.

$$\begin{aligned}
\eta_i &= \log \mu_i, \\
z_i &= \eta_i + (y_i - \mu_i)\frac{\partial \mu_i}{\partial \eta_i} = \eta_i + \frac{y_i - \mu_i}{\mu_i}, \\
w_i &= \frac{1}{V(\mu_i)}\left(\frac{\partial \mu_i}{\partial \eta_i}\right)^2 = \mu_i
\end{aligned}$$

elastic-net 벌점항을 가진 포아송 회귀모형을 만들기 위해 벌점항 $P_\alpha(\boldsymbol{\beta}) = \frac{1}{2}(1-\alpha)||\boldsymbol{\beta}||_2^2 + \alpha||\boldsymbol{\beta}||_1 = \sum_{j=1}^p [\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j|]$ 과 $\lambda > 0$를 추가한다. elastic-net 벌점항이 붙은 포아송 회귀계수의 추정은 다음 형태를 최소화하는 문제와 같다:

$$\min_{(\beta_0, \boldsymbol{\beta}) \in \mathbb{R}^{p+1}} \left[\frac{1}{2n} \sum_{i=1}^n w_i (z_i - \beta_0 - \boldsymbol{x}_i^T \boldsymbol{\beta})^2 + \lambda P_\alpha(\boldsymbol{\beta}) \right]$$

여기서 $\alpha = 0$이면 RIDGE 벌점항, $\alpha = 1$이면 LASSO 벌점항이 된다. 코드에서도 α로 각 벌점항을 나누었으며, elastic-net 벌점항의 경우 $\alpha = 0.5$로 고정하였다. 최적의 λ는 시험 데이터에서의 deviance를 최소로 만드는 λ를 선택하였다. elastic-net의 해를 coordinate descent로 풀면 각 변수 $j = 1, \cdots, p$에 대해 적용하여 수렴할때까지 반복한다:

$$\hat{\beta}_j = \frac{S(\sum_{i=1}^n w_i x_{ij}(z_i - \tilde{z}_i^{(j)}), \lambda\alpha)}{\sum_{i=1}^n w_i x_{ij}^2 + \lambda(1-\alpha)}$$

그리고 intercept β_0는 따로 추정하였으며, 다음의 식으로 추정하였다:

$$\hat{\beta}_0 = \frac{S(\sum_{i=1}^n w_i (z_i - \tilde{z}_i^{(j)}), \lambda\alpha)}{\sum_{i=1}^n w_i + \lambda(1-\alpha)}$$

```r
fit_elastic_poisson_regression = function(X, y, lambda, alpha){
  p = ncol(X)
  sX = scale(X)
  # initialize
  bet = runif(p)
  bet0 = runif(1)

  t = loss = 0
  while(t < 200){
    t = t + 1

    eta = bet0 + sX%*%bet
    mu = exp(eta)
    z = eta + (y-mu)/mu
    w = mu

    bet0 =  sum(w*(z-sX%*%bet))/sum(w)

    for(j in 1:p){
      z_tilda = bet0 + sX[,-j]%*% bet[-j]
      input = sum(w *sX[,j]*(z-z_tilda))
      bet[j] = soft_threshold_elasticnet(input, lambda, alpha)/ (sum(w*sX[,j]^2) +lambda*(1-alpha))

      if(sum(exp(sX %*% bet) == Inf) >=1) break
    }

    muu = exp(bet0 + sX %*% bet) +1e-10
    muu = ifelse(muu == Inf, exp(50), muu)

    loss_new = sum(dpois(y, lambda = muu, log = T)) - sum(abs(bet))

    if(loss_new == Inf | loss_new == -Inf) break
```

```r
    if(abs(loss - loss_new + 1)/abs(loss+1) < 1e-7) break

    loss = loss_new
  } # end while

  return(c(bet0, bet))
}

soft_threshold_elasticnet = function(z, lambda, alpha){
  r = lambda*alpha
  if(z > 0 & r < abs(z)){
    thr = z - r
  } else if (z < 0 & r < abs(z)){
    thr = z + r
  } else if(r > abs(z)){
    thr = 0
  }
  return(thr)
}

find_optimal_lambda = function(train_X, train_y, test_X, test_y, type, lambda, alpha) {
  sX = scale(test_X)
  desX = cbind(1, sX)
  if(type %in% c("lasso", "ridge", "elast")){
    out = matrix(0, length(lambda), 2)
    dev = c()

    for(l in 1:length(lambda)){
      # print(out)
      pred_beta = fit_elastic_poisson_regression(train_X, train_y, lambda = lambda[l], alpha)

      te_mu = exp(desX %*% pred_beta)
      te_mu = ifelse(te_mu == Inf, exp(10)/nrow(test_X), te_mu)

      dev_val = test_y * log((test_y+1)/(te_mu+1)) - (test_y - te_mu)
      dev_val = ifelse(dev_val == Inf, exp(10)/nrow(test_X), dev_val)

      dev[l] = 2 * sum(dev_val)
      out[l,] = c(lambda[l], dev[l])
    }

  }
  if (type == "scad"){
    out = matrix(0, length(lambda), 2)
    dev = c()

    for(l in 1:length(lambda)){
      # print(out)
      pred_beta = fit_scad_poisson_regression(train_X, train_y, lambda = lambda[l], alpha)
      te_mu = exp(desX %*% pred_beta)
      te_mu = ifelse(te_mu == Inf, exp(10)/nrow(test_X), te_mu)

      dev_val = test_y * log((test_y+1)/(te_mu+1)) - (test_y - te_mu)
```

```r
      dev_val = ifelse(dev_val == Inf, exp(10)/nrow(test_X), dev_val)

      dev[l] = 2 * sum(dev_val)
      out[l,] = c(lambda[l], dev[l])
    }
  }

  colnames(out) = c("lambda", "dev")
  out = as.data.frame(out)

  return(out)
}

# glm 코드와 확인
# lasso
fit_elastic_poisson_regression(tr_X, tr_y, lambda=0.3, alpha = 1)
```

```
## [1] 0.9695489 1.0039073 0.9936901 1.0218879
```

```r
glmnet(tr_X, tr_y, family = "poisson", lambda=0.3, alpha = 1)$beta
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##             s0
## V1 0.9119704
## V2 0.9670671
## V3 0.9761763
```

```r
# ridge
fit_elastic_poisson_regression(tr_X, tr_y, lambda=0.3, alpha = 0)
```

```
## [1] 0.9695502 1.0039069 0.9936905 1.0218866
```

```r
glmnet(tr_X, tr_y, family = "poisson", lambda=0.3, alpha = 0)$beta
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##             s0
## V1 0.9148845
## V2 0.9686154
## V3 0.9766168
```

```r
# elasticnet
fit_elastic_poisson_regression(tr_X, tr_y, lambda=0.3, alpha = 0.5)
```

```
## [1] 0.9695495 1.0039071 0.9936903 1.0218872
```

```r
glmnet(tr_X, tr_y, family = "poisson", lambda=0.3, alpha = 0.5)$beta
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##             s0
## V1 0.9134637
## V2 0.9678577
## V3 0.9764023
```

그 다음 SCAD 벌점항을 가진 포아송 회귀모형을 만들기 위해 다음과 같은 벌점항을 추가한다:

$$P(\boldsymbol{\beta}) = \begin{cases} \lambda|\boldsymbol{\beta}|, & \text{if } |\boldsymbol{\beta}| \le \lambda, \\ \frac{2\alpha\lambda|\boldsymbol{\beta}| - |\boldsymbol{\beta}|^2 - \lambda^2}{2(\alpha-1)}, & \text{if } \lambda < |\boldsymbol{\beta}| \le \alpha\lambda, \\ \frac{\lambda^2(\alpha+1)}{2}, & \text{if } |\boldsymbol{\beta}| > \alpha\lambda. \end{cases}$$

여기서 $\lambda \ge 0, \alpha > 2$이다. 그러면 SCAD 회귀계수를 추정하는 것은 working response $z_i = \eta_i + \frac{y_i - \mu_i}{\mu_i}$와 weight $w_i = \mu_i$에 대하여 다음의 형태를 최소화하는 문제와 같다:

$$\min_{(\beta_0, \boldsymbol{\beta}) \in \mathbb{R}^{p+1}} \left[\frac{1}{2n} \sum_{i=1}^{n} w_i (z_i - \beta_0 - \boldsymbol{x}_i^T \boldsymbol{\beta})^2 + P(\boldsymbol{\beta}) \right]$$

그리고 $\tilde{z}_i^{(j)} = X_{-j}\boldsymbol{\beta}_{-j}$로부터 $\tilde{\beta}_j$를 다음과 같이 정의하자:

$$\tilde{\beta}_j = \frac{1}{n} \sum_{i=1}^{n} w_i x_{ij} (z_i - \tilde{z}_i)$$

여기서 X_{-j}와 $\boldsymbol{\beta}_{-j}$는 각각 j번째 변수를 제외한 독립변수들과 회귀계수이다.

이때 α도 파라미터 서치가 필요하지만, 계산의 단순화를 위해 [Fan and Li, 2001]에서 권장하는 값인 3.7로 고정하였다. 최적의 λ는 시험 데이터에서의 deviance를 최소로 만드는 λ를 선택하였다. SCAD의 해는 다음과 같이 정의되며, coordinate descent 알고리즘으로 풀기 위해 각 회귀계수 $j = 1, \cdots, p$에 대해 $\hat{\beta}_j$들이 수렴할때까지 반복한다:

$$\hat{\beta}_j^{SCAD} = \begin{cases} \frac{S(\tilde{\beta}_j, \lambda)}{v_j}, & \text{if } |\tilde{\beta}_j| \le \lambda(v_j + 1), \\ \frac{S(\tilde{\beta}_j, \alpha\lambda/(\alpha-1))}{v_j - 1/(\alpha-1)}, & \text{if } \lambda(v_j+1) < |\tilde{\beta}_j| \le v_j\alpha\lambda, \\ \beta_j^{ols}, & \text{if } |\tilde{\beta}_j| > v_j\alpha\lambda. \end{cases}$$

여기서 $\alpha > 1 + \frac{1}{v_j}$이고 $v_j = \frac{1}{n}\sum_{i=1}^{n} w_i x_{ij}^2$이다.

```
fit_scad_poisson_regression = function(X, y, lambda, alpha){
  p = ncol(X)
  sX = scale(X)
  # initialize
  bet = runif(p)
  bet0 = runif(1)
  desX = cbind(1, sX)

  t = loss = 0
  while(t < 200){
    t = t + 1

    eta = bet0 + sX %*% bet
    mu = exp(eta)
    z = eta + (y-mu)/mu
    w = mu

    bet0 =  sum(w*(z-sX%*%bet))/sum(w)

    for(j in 1:p){
      v = sum(w*sX[,j]^2)
      z_tilde = bet0 + sX[,-j]%*% bet[-j]
      beta_tilde = sum(w *sX[,j]*(z-z_tilde))
      bet[j] = soft_threshold_scad(beta_tilde, v, lambda, alpha=alpha)
```

```r
      if(sum(exp(sX %*% bet) == Inf) >=1) break
    }

    muu = exp(bet0 + sX %*% bet) +1e-10
    muu = ifelse(muu == Inf, exp(50), muu)

    loss_new = sum(dpois(y, lambda = muu, log = T)) - sum(abs(bet))

    if(loss_new == Inf | loss_new == -Inf) break
    if(abs(loss - loss_new + 1)/abs(loss+1) < 1e-7) break

    loss = loss_new
  } # end while

  return(c(bet0, bet))
}

soft_threshold_scad = function(z, v, lambda, alpha){
  if(abs(z) <= lambda*(v+1)){
    thr = soft_threshold_elasticnet(z, lambda, 0)/v
  } else if (lambda*(v+1) < abs(z) & abs(z) <= (v*alpha * lambda)){
    thr = soft_threshold_elasticnet(z, alpha*lambda/(alpha-1), 0)/(v-1/(alpha-1))
  } else if (abs(z) > (v*alpha * lambda)){
    thr = z/v
  }
  return(thr)
}

# glm 코드와 확인
fit_scad_poisson_regression(tr_X, tr_y, lambda = 0.3, alpha = 3.7)
```

```
## [1] 0.9690513 1.0041930 0.9938432 1.0220291
```

```r
ncvreg(tr_X, tr_y, family = 'poisson', penalty="SCAD", alpha = 3.7)$beta
```

```
## Warning in ncvreg(tr_X, tr_y, family = "poisson", penalty = "SCAD", alpha =
## 3.7): Maximum number of iterations reached

##                3.4140     3.1839     2.9693     2.7692     2.5825     2.4085
## (Intercept) 2.441129  2.4137260  2.4097095  2.4050618  2.3996917  2.3934503
## V1          0.000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## V2          0.000000 -0.1367718 -0.1476389 -0.1593981 -0.1721280 -0.1858643
## V3          0.000000 -0.1437135 -0.1553087 -0.1678825 -0.1815238 -0.1962763
##                 2.2462     2.0948     1.9536     1.8219     1.6991     1.5846
## (Intercept)  2.3862425  2.3778826  2.3595594  2.3465588  2.3315227  2.3140766
## V1           0.0000000  0.0000000 -0.1377925 -0.1493844 -0.1620912 -0.1759845
## V2          -0.2007125 -0.2167039 -0.2331559 -0.2516603 -0.2715857 -0.2929103
## V3          -0.2122574 -0.2295047 -0.2503868 -0.2707741 -0.2927437 -0.3162538
##                 1.4778
## (Intercept)  2.2485406
## V1          -0.1614124
## V2          -0.2552440
## V3          -0.2680883
```

3. 실험의 평가

실험의 평가는 크게 회귀계수 추정의 정확성과 변수선택의 성능, 그리고 컴퓨팅 시간의 측정으로 한다.

먼저 회귀계수 추정의 정확성은 MSE, Variance, Bias의 세 가지 측도로 평가하였으며, $k = 1, \cdots, N = 100$번의 반복에 대해 각 측정값은 다음과 같이 계산한다:

- MSE: $\frac{1}{N}\sum_{k=1}^{N}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^T(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) = \frac{1}{N}\sum_{k=1}^{N}\sum_{j=1}^{p}(\hat{\beta}_{jk} - \beta_{jk})^2$
- Variance: $tr\{\frac{1}{N}\sum_{k=1}^{N}(\hat{\boldsymbol{\beta}}_j - \bar{\boldsymbol{\beta}})^T(\hat{\boldsymbol{\beta}}_j - \bar{\boldsymbol{\beta}})\} = \sum_{j=1}^{p}\frac{1}{N}\sum_{k=1}^{N}(\hat{\beta}_{jk} - \bar{\beta}_j)^2$
- Bias: $\mathbf{1}^T|\bar{\boldsymbol{\beta}} - \boldsymbol{\beta}| = \sum_{j=1}^{p}|\bar{\beta}_j - \beta_j|$

그 다음 변수선택의 성능은 CS (옳게 선택된 변수의 갯수), IS (잘못 선택된 변수의 갯수), AC (변수선택 결과가 참과 같으면 1, 그렇지 않으면 0)의 세 가지 측도로 평가하였으며, 컴퓨팅 시간은 `system.time`함수로 측정하였다. 아래 코드는 평가 측도를 계산하기 위해 구현한 코드이다.

```
return_AC = function(true, pois, ridge, lasso, elast, scad){

  pois_NK = do.call("cbind", pois)
  ridge_NK = do.call("cbind", ridge)
  lasso_NK = do.call("cbind", lasso)
  elast_NK = do.call("cbind", elast)
  scad_NK = do.call("cbind", scad)

  pois_NK_CS = ifelse(pois_NK > 0, 1, 0)
  ridge_NK_CS = ifelse(ridge_NK > 0, 1, 0)
  lasso_NK_CS = ifelse(lasso_NK > 0, 1, 0)
  elast_NK_CS = ifelse(elast_NK > 0, 1, 0)
  scad_NK_CS = ifelse(scad_NK > 0, 1, 0)

  p = nrow(pois_NK_CS)
  pois_CS = sapply(1:N_rep, function(j) sum(pois_NK_CS[,j] == true) == p)
  ridge_CS = sapply(1:N_rep, function(j) sum(ridge_NK_CS[,j] == true) == p)
  lasso_CS = sapply(1:N_rep, function(j) sum(lasso_NK_CS[,j] == true) == p)
  elast_CS = sapply(1:N_rep, function(j) sum(elast_NK_CS[,j] == true) == p)
  scad_CS = sapply(1:N_rep, function(j) sum(scad_NK_CS[,j] == true) == p)

  pois_CS = ifelse(pois_CS, 1, 0)
  ridge_CS = ifelse(ridge_CS, 1, 0)
  lasso_CS = ifelse(lasso_CS, 1, 0)
  elast_CS = ifelse(elast_CS, 1, 0)
  scad_CS = ifelse(scad_CS, 1, 0)

  return(list('mean' = c(sum(pois_CS), sum(ridge_CS), sum(lasso_CS), sum(elast_CS), sum(scad_CS)),
              'se' = c(sd(pois_CS), sd(ridge_CS), sd(lasso_CS), sd(elast_CS), sd(scad_CS))/sqrt(N_rep)))
}

return_IS = function(true, pois, ridge, lasso, elast, scad){
  pois_NK = do.call("cbind", pois)
  ridge_NK = do.call("cbind", ridge)
  lasso_NK = do.call("cbind", lasso)
  elast_NK = do.call("cbind", elast)
  scad_NK = do.call("cbind", scad)

  pois_NK_CS = ifelse(pois_NK > 0, 1, 0)
  ridge_NK_CS = ifelse(ridge_NK > 0, 1, 0)
```

```r
  lasso_NK_CS = ifelse(lasso_NK > 0, 1, 0)
  elast_NK_CS = ifelse(elast_NK > 0, 1, 0)
  scad_NK_CS = ifelse(scad_NK > 0, 1, 0)

  pois_CS = sapply(1:N_rep, function(j) sum(pois_NK_CS[,j] != true))
  ridge_CS = sapply(1:N_rep, function(j) sum(ridge_NK_CS[,j] != true))
  lasso_CS = sapply(1:N_rep, function(j) sum(lasso_NK_CS[,j] != true))
  elast_CS = sapply(1:N_rep, function(j) sum(elast_NK_CS[,j] != true))
  scad_CS = sapply(1:N_rep, function(j) sum(scad_NK_CS[,j] != true))

  return(list('mean' = c(mean(pois_CS), mean(ridge_CS), mean(lasso_CS), mean(elast_CS), mean(scad_CS)),
              'se' = c(sd(pois_CS), sd(ridge_CS), sd(lasso_CS), sd(elast_CS), sd(scad_CS))/sqrt(N_rep))
}

return_CS = function(true, pois, ridge, lasso, elast, scad){
  pois_NK = do.call("cbind", pois)
  ridge_NK = do.call("cbind", ridge)
  lasso_NK = do.call("cbind", lasso)
  elast_NK = do.call("cbind", elast)
  scad_NK = do.call("cbind", scad)

  pois_NK_CS = ifelse(pois_NK > 0, 1, 0)
  ridge_NK_CS = ifelse(ridge_NK > 0, 1, 0)
  lasso_NK_CS = ifelse(lasso_NK > 0, 1, 0)
  elast_NK_CS = ifelse(elast_NK > 0, 1, 0)
  scad_NK_CS = ifelse(scad_NK > 0, 1, 0)

  pois_CS = sapply(1:N_rep, function(j) sum(pois_NK_CS[,j] == true))
  ridge_CS = sapply(1:N_rep, function(j) sum(ridge_NK_CS[,j] == true))
  lasso_CS = sapply(1:N_rep, function(j) sum(lasso_NK_CS[,j] == true))
  elast_CS = sapply(1:N_rep, function(j) sum(elast_NK_CS[,j] == true))
  scad_CS = sapply(1:N_rep, function(j) sum(scad_NK_CS[,j] == true))

  return(list('mean' = c(mean(pois_CS), mean(ridge_CS), mean(lasso_CS), mean(elast_CS), mean(scad_CS)),
              'se' = c(sd(pois_CS), sd(ridge_CS), sd(lasso_CS), sd(elast_CS), sd(scad_CS))/sqrt(N_rep))
}

return_BIAS = function(true, pois, ridge, lasso, elast, scad){
  pois_NK = do.call("cbind", pois)
  pois_bark = rowMeans(pois_NK)

  ridge_NK = do.call("cbind", ridge)
  ridge_bark = rowMeans(ridge_NK)

  lasso_NK = do.call("cbind", lasso)
  lasso_bark = rowMeans(lasso_NK)

  elast_NK = do.call("cbind", elast)
  elast_bark = rowMeans(elast_NK)

  scad_NK = do.call("cbind", scad)
  scad_bark = rowMeans(scad_NK)
```

```
  pois_BIAS = abs(true - pois_bark)
  ridge_BIAS = abs(true - ridge_bark)
  lasso_BIAS = abs(true - lasso_bark)
  elast_BIAS = abs(true - elast_bark)
  scad_BIAS = abs(true - scad_bark)

  return(list('mean' = c(sum(pois_BIAS), sum(ridge_BIAS), sum(lasso_BIAS), sum(elast_BIAS), sum(scad_BI
              'se' = c(sd(pois_BIAS), sd(ridge_BIAS), sd(lasso_BIAS), sd(elast_BIAS), sd(scad_BIAS))))
}

return_VAR = function(pois, ridge, lasso, elast, scad){
  pois_NK = do.call("cbind", pois)
  pois_bark = rowMeans(pois_NK)

  ridge_NK = do.call("cbind", ridge)
  ridge_bark = rowMeans(ridge_NK)

  lasso_NK = do.call("cbind", lasso)
  lasso_bark = rowMeans(lasso_NK)

  elast_NK = do.call("cbind", elast)
  elast_bark = rowMeans(elast_NK)

  scad_NK = do.call("cbind", scad)
  scad_bark = rowMeans(scad_NK)

  p = length(pois_bark)
  pois_VARm = sum( sapply(1:p, function(j) {mean((pois_NK[j,] - pois_bark[j])^2)}) )
  ridge_VARm = sum( sapply(1:p, function(j) {mean((ridge_NK[j,] - ridge_bark[j])^2)}) )
  lasso_VARm = sum( sapply(1:p, function(j) {mean((lasso_NK[j,] - lasso_bark[j])^2)}) )
  elast_VARm = sum( sapply(1:p, function(j) {mean((elast_NK[j,] - elast_bark[j])^2)}) )
  scad_VARm = sum( sapply(1:p, function(j) {mean((scad_NK[j,] - scad_bark[j])^2)}) )

  pois_VARs = sum( sapply(1:p, function(j) {sd((pois_NK[j,] - pois_bark[j])^2)/sqrt(N_rep)}) )
  ridge_VARs = sum( sapply(1:p, function(j) {sd((ridge_NK[j,] - ridge_bark[j])^2)/sqrt(N_rep)}) )
  lasso_VARs = sum( sapply(1:p, function(j) {sd((lasso_NK[j,] - lasso_bark[j])^2)/sqrt(N_rep)}) )
  elast_VARs = sum( sapply(1:p, function(j) {sd((elast_NK[j,] - elast_bark[j])^2/sqrt(N_rep))}) )
  scad_VARs = sum( sapply(1:p, function(j) {sd((scad_NK[j,] - scad_bark[j])^2)/sqrt(N_rep)}) )

  return(list('mean' = c(pois_VARm, ridge_VARm, lasso_VARm, elast_VARm, scad_VARm),
              'se' = c(pois_VARs, ridge_VARs, lasso_VARs, elast_VARs, scad_VARs)))
}


return_MSE = function(true, pois, ridge, lasso, elast, scad){
  pois_iter = lapply(1:N_rep, function(j) {sum((pois[[j]] - true)^2)})
  ridge_iter = lapply(1:N_rep, function(j) {sum((ridge[[j]] - true)^2)})
  lasso_iter = lapply(1:N_rep, function(j) {sum((lasso[[j]] - true)^2)})
  elast_iter = lapply(1:N_rep, function(j) {sum((elast[[j]] - true)^2)})
  scad_iter = lapply(1:N_rep, function(j) {sum((scad[[j]] - true)^2)})

  pois_mse = do.call("c", pois_iter)
  ridge_mse = do.call("c", ridge_iter)
```

```r
  lasso_mse = do.call("c", lasso_iter)
  elast_mse = do.call("c", elast_iter)
  scad_mse = do.call("c", scad_iter)

  return(list('mean' = c(mean(pois_mse), mean(ridge_mse), mean(lasso_mse), mean(elast_mse), mean(scad_ms
              'se' = c(sd(pois_mse), sd(ridge_mse), sd(lasso_mse), sd(elast_mse), sd(scad_mse))/sqrt(N_r
}

result_glm_compare = function(pois, ridge, lasso, elast, scad, glm_pois, glm_ridge, glm_lasso, glm_elast
  compare_pois = sapply(1:N_rep, function(j) sum(abs(glm_pois[[j]] - pois[[j]])))
  compare_ridge = sapply(1:N_rep, function(j) sum(abs(glm_ridge[[j]] - ridge[[j]])))
  compare_lasso = sapply(1:N_rep, function(j) sum(abs(glm_lasso[[j]] - lasso[[j]])))
  compare_elast = sapply(1:N_rep, function(j) sum(abs(glm_elast[[j]] - elast[[j]])))
  compare_scad = sapply(1:N_rep, function(j) sum(abs(glm_scad[[j]] - scad[[j]])))

  return(list('mean' = c(mean(compare_pois), mean(compare_ridge), mean(compare_lasso), mean(compare_elas
              'se' = c(sd(compare_pois), sd(compare_ridge), sd(compare_lasso), sd(compare_elast), sd(com
}
```

4. 실행

위의 코드들은 아래의 코드로 실행할 수 있다.

```r
# N_rep = 100  # 반복수
# lambda_grid = seq(0.001, 2, length.out = 20)
#
# MSEm = VARm = BIASm = MSEs = VARs = BIASs = matrix(0, 4, 5)
# CSm = ISm = ACm = CSs = ISs = ACs = timem = times = matrix(0, 4, 5)
# MAEm = MAEs = matrix(0, 4, 5)
#
# k = 0
# set.seed(2023020358)
# for(pp in c(3, 50)){
#   for(rr in c(0, 0.7)){
#     k = k + 1
#     print(k)
#     fit_pois = fit_ridge = fit_lasso = fit_elast = fit_scad = list()
#     glm_pois = glm_ridge = glm_lasso = glm_elast = glm_scad = list()
#     time = matrix(0, N_rep, 5)
#     for(iter in 1:N_rep){
#       cat(iter)
#       sim_dat = sim_data(train_N = 500, test_N = 5000, p = pp, rho = rr)  # 데이터 생성
#       true_beta = c(1, sim_dat$beta)
#
#       tr_X = sim_dat$train_X
#       tr_y = sim_dat$train_y
#       te_X = sim_dat$test_X
#       te_y = sim_dat$test_y
#
#       # 1. POIS
#       t1 = system.time({
#         fit_pois[[iter]] = fit_poisson_resgression(tr_X, tr_y)$beta
#       })[3]
#       glm_pois[[iter]] = glm(tr_y~tr_X, family="poisson")$coefficient %>% as.vector
```

```
#
#       # 2. ridge
#       t2 = system.time({
#         find_ridge_lambda = find_optimal_lambda(tr_X, tr_y, te_X, te_y, type = "ridge", lambda_grid, a
#         opt_lambda = lambda_grid[which.min(find_ridge_lambda[,2])]
#         fit_ridge[[iter]] = fit_elastic_poisson_regression(tr_X, tr_y, lambda = opt_lambda, alpha = 0)
#       })[3]
#
#       glm_cv_ridge = glmnet(tr_X, tr_y,family="poisson",alpha = 0)
#       opt_lambda = lambda_grid[which.min(glm_cv_ridge$dev.ratio)]
#       glm_ridge[[iter]] = glmnet(tr_X, tr_y,family="poisson",lambda = opt_lambda, alpha = 0)$beta%>% a
#
#       # 3. lasso
#       t3 = system.time({
#         find_lasso_lambda = find_optimal_lambda(tr_X, tr_y, te_X, te_y, type = "lasso", lambda_grid, a
#         opt_lambda = lambda_grid[which.min(find_lasso_lambda[,2])]
#         fit_lasso[[iter]] = fit_elastic_poisson_regression(tr_X, tr_y, lambda = opt_lambda, alpha = 1)
#       })[3]
#
#       glm_cv_lasso = glmnet(tr_X, tr_y,family="poisson",alpha = 1)
#       opt_lambda = lambda_grid[which.min(glm_cv_lasso$dev.ratio)]
#       glm_lasso[[iter]] = glmnet(tr_X, tr_y,family="poisson",lambda = opt_lambda, alpha = 1)$beta%>% a
#
#       # 4. elast
#       t4 = system.time({
#         find_elast_lambda = find_optimal_lambda(tr_X, tr_y, te_X, te_y, type = "elast", lambda_grid, a
#         opt_lambda = lambda_grid[which.min(find_elast_lambda[,2])]
#         fit_elast[[iter]] = fit_elastic_poisson_regression(tr_X, tr_y, lambda = opt_lambda, alpha = 0
#       })[3]
#
#       glm_cv_elast = glmnet(tr_X, tr_y, family="poisson", alpha = 0.5)
#       opt_lambda = lambda_grid[which.min(glm_cv_elast$dev.ratio)]
#       glm_elast[[iter]] = glmnet(tr_X, tr_y, family="poisson", lambda = opt_lambda, alpha = 0.5)$beta
#
#       # 5. scad
#       t5 = system.time({
#         find_scad_lambda = find_optimal_lambda(tr_X, tr_y, te_X, te_y, type = "scad", lambda_grid, al
#         opt_lambda = lambda_grid[which.min(find_scad_lambda[,2])]
#         fit_scad[[iter]] = fit_scad_poisson_regression(tr_X, tr_y, lambda = opt_lambda, alpha = 3.7)
#       })[3]
#
#       glm_cv_scad = ncvreg(tr_X, tr_y, family = 'poisson', penalty="SCAD", alpha = 3.7)
#       glm_scad[[iter]] = glm_cv_scad$beta[,which.min(glm_cv_scad$loss)]
#
#       time[iter,] = c(t1, t2, t3, t4, t5)
#     }
#
#     # perfomance
#     MSE_result = return_MSE(true_beta, fit_pois, fit_ridge, fit_lasso, fit_elast, fit_scad)
#     VAR_result = return_VAR(fit_pois, fit_ridge, fit_lasso, fit_elast, fit_scad)
#     BIAS_result = return_BIAS(true_beta, fit_pois, fit_ridge, fit_lasso, fit_elast, fit_scad)
#
#     MSEm[k,] = MSE_result$mean
```

```
#     VARm[k,] = VAR_result$mean
#     BIASm[k,] = BIAS_result$mean
#     MSEs[k,] = MSE_result$se
#     VARs[k,] = VAR_result$se
#     BIASs[k,] = BIAS_result$se
#
#     CS_result = return_CS(true_beta, fit_pois, fit_ridge, fit_lasso, fit_elast, fit_scad)
#     IS_result = return_IS(true_beta, fit_pois, fit_ridge, fit_lasso, fit_elast, fit_scad)
#     AC_result = return_AC(true_beta, fit_pois, fit_ridge, fit_lasso, fit_elast, fit_scad)
#
#     CSm[k,] = CS_result$mean
#     ISm[k,] = IS_result$mean
#     ACm[k,] = AC_result$mean
#     CSs[k,] = CS_result$se
#     ISs[k,] = IS_result$se
#     ACs[k,] = AC_result$se
#
#     timem[k,] = colMeans(time)
#     times[k,] = apply(time, 2, sd)/sqrt(N_rep)
#
#     # glm과 비교
#     MAE_result = result_glm_compare(fit_pois, fit_ridge, fit_lasso, fit_elast, fit_scad,
#                                     glm_pois, glm_ridge, glm_lasso, glm_elast, glm_scad)
#     MAEm[k,] = MAE_result$mean
#     MAEs[k,] = MAE_result$se
#   } # for rho
# } # for p
```


[1] Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the American statistical Association, 96(456), 1348-1360.

[2] Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. Journal of statistical software, 33(1), 1.

[3] Breheny, P., & Huang, J. (2011). Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. The annals of applied statistics, 5(1), 232.