ST509_HW3_2024020409

Hwijun Kwon

2024-03-29

3.

By R

```
nr_poisson <- function(X, y, init = NULL, max_iter = 1000, eps= 1.0e-5) {
  if (is.null(init)) init = rep(0, ncol(X))
  beta = init
  iter = 0
  for (i in 1:max_iter) {
    eta <- X %*% beta # eta = X*beta
    mu <- exp(eta)
    w <- diag(c(mu)) # W = diag(Var(Y_i)) = diag(mu_i)

    z = X %*% beta + diag(c(1/mu)) %*% (y-mu) # z_t = X*beta_t + (y-mu_t)/W_t
    # X(T)X*beta = X(T)y

    X_tilde = diag(c(mu**(1/2))) %*% X  # X(~) = W^(1/2) * X
    z_tilde = diag(c(mu**(1/2))) %*% z  # z(~) = W^(1/2) * z
    qr.obj = qr(X_tilde)
    new_beta = backsolve(qr.obj$qr, qr.qty(qr.obj, z_tilde))

    if (max(abs(new_beta -beta))/max(abs(beta)) < eps) {
      message("Convergence reached after ", i, " iterations.")
      break
    }
    beta <- new_beta
    iter = iter + 1
  }
  if (iter == max_iter) warning("may not be converged")
  obj <- list(est = c(beta), iterations = iter)
}
```

Experiment with random sample

```
set.seed(1)
n = 100 ; p = 3

x <- matrix(rnorm(n * p), n , p)
X <- cbind(rep(1, n ), x)
beta = rep(1, p+1)
eta = X %*% beta
mu = exp(eta)
```

```r
y = rpois(n, mu)

obj1 <- nr_poisson(X, y, init = NULL, max_iter = 1000)
```

```
## Convergence reached after 34 iterations.
```

```r
obj2 <- glm(y ~ x, family = poisson(link = 'log'))

hat_beta1 = obj1$est
hat_beta2 = coefficients(obj2)

print(head(cbind(hat_beta1, hat_beta2)))
```

```
##             hat_beta1 hat_beta2
## (Intercept) 1.0081858 1.0081857
## x1          1.0226925 1.0226927
## x2          1.0059156 1.0059155
## x3          0.9749594 0.9749591
```

By Python

```python
import numpy as np
def nr_pois(X, y, init = np.zeros(5), max_iter=1000, eps=1.0e-5):
    n, p = X.shape
    beta = np.zeros(p)
    for i in range(max_iter):
        eta = X @ beta
        mu = np.exp(eta)
        grad = X.T @ (y - mu)
        W = np.diag(mu)
        H = -(X.T @ W @ X)
        beta_new = beta - np.linalg.inv(H) @ grad
        if np.sum(np.abs(beta_new - beta)) < eps:
            break
        beta = beta_new
    return beta
```

Poisson Regression with NR Method

```python
import numpy as np
import statsmodels.api as sm
from scipy.stats import poisson

# Random Sample Generating
np.random.seed(1)
n, p = 100, 3
x = np.random.normal(0, 1, (n, p))
X = np.c_[np.ones(n), x] # Design Matrix for X
beta = np.ones(p + 1)

eta = X.dot(beta)
mu = np.exp(eta)
y = poisson.rvs(mu)
```

```python
# sm.GLM vs nr_pois

poisson_model = sm.GLM(y, X, family=sm.families.Poisson()).fit()
print(poisson_model.summary())
```

Experiment with Random Sample

```
##                  Generalized Linear Model Regression Results
## ==============================================================================
## Dep. Variable:                      y   No. Observations:                  100
## Model:                            GLM   Df Residuals:                       96
## Model Family:                 Poisson   Df Model:                            3
## Link Function:                    Log   Scale:                          1.0000
## Method:                          IRLS   Log-Likelihood:                -193.26
## Date:                Mon, 01 Apr 2024   Deviance:                       97.744
## Time:                        21:22:01   Pearson chi2:                     89.2
## No. Iterations:                     6   Pseudo R-squ. (CS):              1.000
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          z      P>|z|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const          0.9739      0.059     16.594      0.000       0.859       1.089
## x1             0.9782      0.038     25.429      0.000       0.903       1.054
## x2             0.9880      0.035     28.610      0.000       0.920       1.056
## x3             1.0169      0.031     32.574      0.000       0.956       1.078
## ==============================================================================
```

```python
nr_pois(X, y)
```

```
## array([0.97390604, 0.97820204, 0.98799619, 1.01688849])
```

4. Smoking Data

By R

```r
data = read.table("/Users/hj/dropbox/smoking.dat", fill = TRUE, header = FALSE)
df = data[2:nrow(data), 2:ncol(data), drop = FALSE]

colnames(df) = c('age', 'smoke', 'pop', 'dead')
rownames(df) = 1:nrow(df)
```

```r
# age : 80+ => 80-90, subsitute by median
df$age = gsub("80\\+", "80-90", df$age)
df$age <- sapply(df$age, function(x) {
  ages <- as.numeric(unlist(strsplit(x, "-")))
  mean(ages)
})
# smoke
df$smoke = as.integer(factor(df$smoke), levels = unique(df$smoke))
# pop
df$pop = as.integer(df$pop)
# rate : Death per 1000 people
df$rate = as.integer(1000* df$dead/df$pop)
```

```r
df_matrix = as.matrix(df)
head(df_matrix)
```

Data Preprocessing

```
##   age smoke  pop dead rate
## 1  42     4  656   18   27
## 2  52     4  359   22   61
## 3  52     4  249   19   76
## 4  57     4  632   55   87
## 5  62     4 1067  117  109
## 6  67     4  897  170  189
```

```r
x = df_matrix[,c("age", "smoke")]
X <- cbind(rep(1, n ), x) # design matrix for X
```

Parameter Estimation

```
## Warning in cbind(rep(1, n), x): number of rows of result is not a multiple of
## vector length (arg 1)
```

```r
y = df_matrix[,c("rate")]
#obj1 <- nr_poisson(X, y, init = NULL, max_iter = 1000)
obj1 <- nr_poisson(X, y,  max_iter = 1000)
```

```
## Convergence reached after 261 iterations.
```

```r
obj2 <- glm(y ~ x, family = poisson(link = 'log'))

hat_beta1 = obj1$est
hat_beta2 = coefficients(obj2)

print(head(cbind(hat_beta1, hat_beta2)))
```

```
##               hat_beta1   hat_beta2
## (Intercept)  1.29266363  1.29266350
## xage         0.06007055  0.06007055
## xsmoke      -0.01860394 -0.01860395
```

By Python

```python
import pandas as pd

file_path = "/Users/hj/dropbox/smoking.dat"
df = pd.read_csv(file_path, sep='\s+')

# age : subsitituion with average age
df["age"] = df["age"].replace('80+', '80-84')
df["age"] = df["age"].apply(lambda x : sum(int(n) for n in x.split('-'))/2)

# rate : death with 1000
df["rate"] = 1000* df["dead"]/df["pop"]
df['rate'] = df['rate'].astype(int)
```

```python
# smoke : dummy variable
smoke_dummies = pd.get_dummies(df['smoke'], prefix='smoke')
df_encoded = pd.concat([df, smoke_dummies], axis=1)
```

Data Preprocessing