Project for P-spline and Multilevel

Choi TaeYoung

2020-08-06

Contents

1 필요한 패키지

2 데이터

- Y data : Y데이터의 경우 28주 (2009년 1월 ~ 2018년 12월) 동안의 유튜브 Hot200 진입 횟수를 나타내었다.
- X data : Top 50 유튜버의 구독자수를 나타내었다.

3 데이터 정리 및 Goodness of fit test를 통한 적절한 모델 찾기

- X, Y 데이터 모두 리스트화를 거쳤다. Y데이터가 허들모델이라는 가정으로 각 행마다 0이 얼마나 포함되어 있는지 알아보았다.
- 그 결과 2, 14, 48행 이외에는 0을 포함하지 않아서 허들모델이나 zero inflated 방법을 이용하여 모델을 적합할 수 없었다.
- 그래서 우리는 Goodness of Fit(GoF) 를 이용하여 일반화 선형모형의 적합도를 검정해보았다.

```
x_list <- x_gdp[,-1] %>% as.data.frame() %>% unlist() %>% as.list()

#How many zero in y_list?
y_list <- obs_y[,-1] %>% t() %>% as.data.frame()
y_zero <- NULL
for(m in 1:28){
  zero <- NULL
  zero <- length(which(y_list[m] == 0))/50
  y_zero <- rbind(y_zero,zero)

  zero_count <- length(which(y_zero > 0))
  zero_where <- which(y_zero > 0)
  zero_count
  zero_where
}
zero_count
```

```
## [1] 28
```

3.1 GoF 결과

- 그 결과 포아송 GoF는 모두 0으로 나왔으며, 음이항분포 GoF는 낮은 값을 보였다. 즉, 포아송분포를 사용하였을 때 과대산포가 발생하므로, 음이항분포를 이용하여 모형적합을 시도했다.

```
#GOF Calculate
goodness <- NULL
  for(m in 1:28){
    result1_out <- NULL
    result2_out <- NULL
    results1 <- glm(unlist(y_list[m]) ~ unlist(x_list), family = poisson, maxit=1000)
    results2 <- glm.nb(unlist(y_list[m]) ~ unlist(x_list), maxit=1000)


    poi_GOF <- 1 - pchisq(summary(results1)$deviance,
           summary(results1)$df.residual
           )
    nb_GOF <- 1 - pchisq(summary(results2)$deviance,
           summary(results2)$df.residual
           )
    out <- cbind(poi_GOF,nb_GOF)
    goodness <- rbind(goodness, out)
  }

tail(goodness)
```

```
##            poi_GOF    nb_GOF
## [23,] 2.689082e-11 0.2208299
## [24,] 7.321759e-10 0.1360174
## [25,] 5.762351e-07 0.1543787
## [26,] 0.000000e+00 0.4561886
## [27,] 0.000000e+00 0.2433231
## [28,] 0.000000e+00 0.3198993
```

4 Multilevel 모델에 적용

- 논문의 방법인 EM알고리즘을 통해 multilevel spline 방법으로 최적의 μ 벡터를 찾았다.

```
x_list <- x_gdp[,-1] %>% as.data.frame() %>% unlist() %>% as.list()
y_list <- obs_y[,-1] %>% t() %>% as.data.frame()
#multilevel

  #beta_hat_vector 구하기

  grain_out <- NULL
  J=28
  beta_hat <- NULL
  for(m in 1:28){
    result2_out <- NULL
    results2 <- glm(unlist(y_list[m]) ~ unlist(x_list), family = poisson, maxit=1000)
      kth_beta_hat <- coef(results2)[2]
      kth_var <- diag(vcov(results2))[2]
      grain_out <- list(kth_beta_hat, kth_var)
      grain_out
    beta_hat <- rbind(beta_hat,grain_out)
```

```
  }
```

- p-spline 기법을 활용하여 새롭게 짠 코드로 리얼데이터에 적용

```
lambda <- c(1e-11,1e-10,1e-09,1e-08,1e-07,1e-06,1e-05,1e-04,1e-03,1e-02,0.1,1, 10,100)
GCV_vec <- NULL

n <- length(unlist(beta_hat[,1]))
IK <- unlist(beta_hat[,1])[-c(1, n)]
EK <- unlist(beta_hat[,1])[c(1, n)]
B = GetBSpline(unlist(beta_hat[,1]), deg = 3, IK, EK)

for(i in 1:length(lambda)){
EM_out <- main_EM_p(beta_hat_vec = unlist(beta_hat[,1]), V = diag(unlist(beta_hat[,2])),
                    B=GetBSpline(unlist(beta_hat[,1]), deg = 3, IK, EK), D=GetDiffMatrix(dim(B)[1], 2),
GCV_vec <- rbind(GCV_vec,EM_out$GCV)
}
```

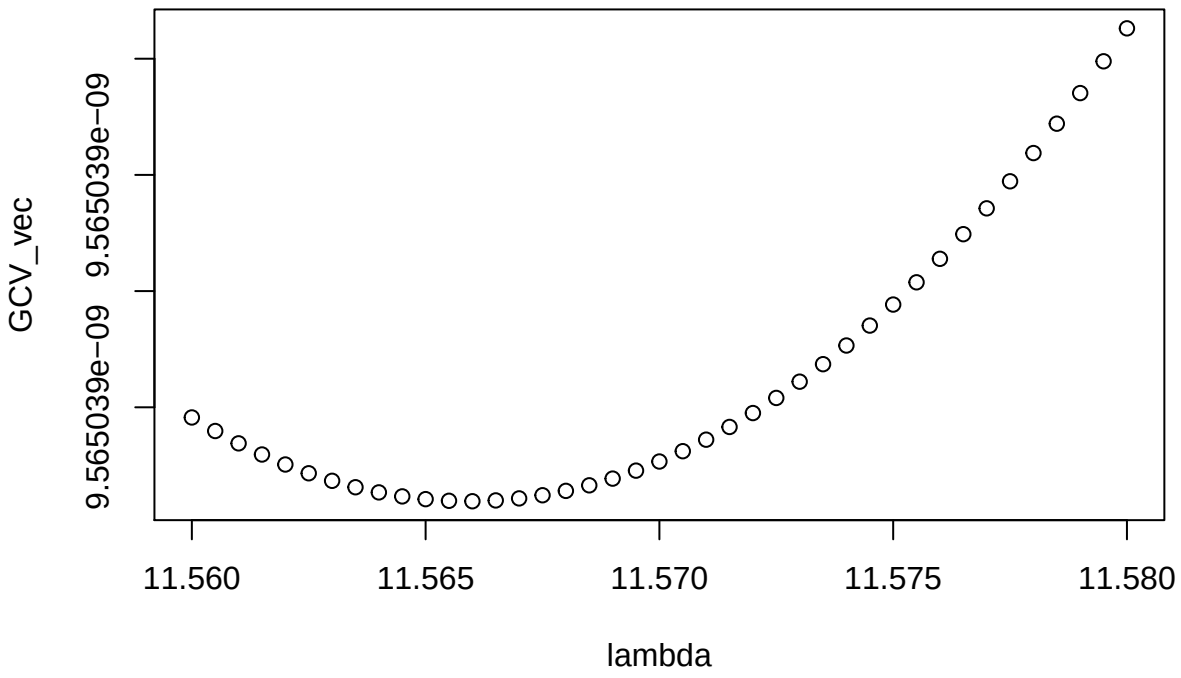
```
## for lambda = 1e-11 max iteration reached; may need to double check
## for lambda = 1e-10 max iteration reached; may need to double check
## for lambda = 1e-09 max iteration reached; may need to double check
```

```
lambda[which.min(GCV_vec)]
```

```
## [1] 10
```

- lambda[which.min(GCV_vec)]을 실행할 때, 100이 나온다.
- 그래서 100근처에서 GCV벡터를 더 찾아보기로 한다.

```
lambda <- seq(11.56, 11.58, by=.0005)
GCV_vec <- NULL

for(i in 1:length(lambda)){
EM_out <- main_EM_p(beta_hat_vec = unlist(beta_hat[,1]), V = diag(unlist(beta_hat[,2])),
                    B=GetBSpline(unlist(beta_hat[,1]), deg = 3, IK, EK), D=GetDiffMatrix(dim(B)[1], 2),
GCV_vec <- rbind(GCV_vec,EM_out$GCV)
}


plot(lambda, GCV_vec)
```



- 최적의 GCV_vec로 EM_out구하기 <- mu_hat 구함

```
lambda[which.min(GCV_vec)]
```

```
## [1] 11.566
```

```
EM_out <- main_EM_p(beta_hat_vec = unlist(beta_hat[,1]), V = diag(unlist(beta_hat[,2])),
                    B=GetBSpline(unlist(beta_hat[,1]), deg = 3, IK, EK), D=GetDiffMatrix(dim(B)[1], 2),
tail(EM_out$mu)
```

```
##                [,1]
## [23,]  2.514895e-05
## [24,] -3.154945e-04
## [25,] -1.159098e-04
## [26,] -4.372859e-04
## [27,]  5.925157e-05
## [28,] -2.384524e-04
```

- Multilevel과 성능을 비교하기위해서 Naive한 방법으로 구해보자.
- Naive기법 역시 P-spline으로 코드를 짠 후 실행했다.

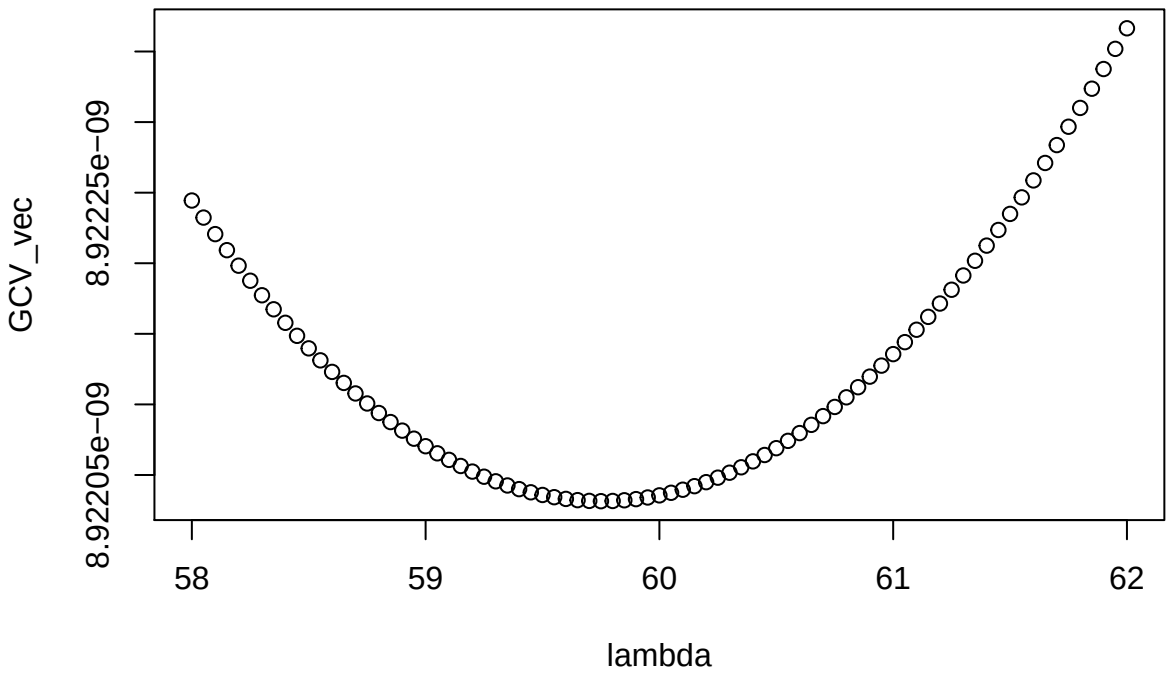
```
#naive
GCV_vec <- NULL
lambda <- c(1e-10,1e-09,1e-08,1e-07,1e-06,1e-05,1e-04,1e-03,1e-02,0.1,1,10,100,1000, 10^4, 10^5, 10^6)
for(i in 1:length(lambda)){
  naive_out <- naive_ss_p(beta_hat_vec = unlist(beta_hat[,1]), B=GetBSpline(unlist(beta_hat[,1]), deg =
  GCV_vec <- rbind(GCV_vec,naive_out$GCV)
}

lambda[which.min(GCV_vec)]
```

```
## [1] 100
```

4.1 Naive's GCV vector 찾기

- Naive 역시 비슷한 방법으로 풀어나간다.

```
GCV_vec <- NULL
lambda <- seq(58, 62, by=.05)
for(i in 1:length(lambda)){
  naive_out <- naive_ss_p(beta_hat_vec = unlist(beta_hat[,1]), lambda = lambda[i],
                          B=GetBSpline(unlist(beta_hat[,1]), deg = 3, IK, EK), D=GetDiffMatrix(dim(B)[1]
  GCV_vec <- rbind(GCV_vec,naive_out$GCV)
}
plot(lambda, GCV_vec)
```



```
lambda[which.min(GCV_vec)]
```

```
## [1] 59.75
```

```
naive_out <- naive_ss_p(beta_hat_vec = unlist(beta_hat[,1]), B=GetBSpline(unlist(beta_hat[,1]), deg = 3


tail(naive_out$mu)
```

```
##                 [,1]
## [23,]  1.138370e-05
## [24,] -3.347248e-04
## [25,] -9.044592e-05
## [26,] -4.735585e-06
## [27,]  2.064792e-05
## [28,] -1.036406e-04
```

5 그래프

```
# hat_all
single_beta <- unlist(beta_hat[,1]) %>% as.vector()
mu_z_naive <- naive_out$mu %>% as.vector()
mu_z_multi <- EM_out$mu %>% as.vector()

hat_all <- cbind(mu_z_naive,mu_z_multi,single_beta) %>% as.data.frame

test_mon <- fread("y_yt_week.csv")
test_mon <- test_mon[-1,1]
hat_all <- cbind(test_mon,hat_all)
```

```
## Warning in as.data.table.list(x, keep.rownames = keep.rownames, check.names
## = check.names, : Item 1 has 27 rows but longest item has 28; recycled with
## remainder.
```

```
hat_all <- rename(hat_all, Week = V1)

hat_all$Week <- parse_date_time(hat_all$Week, "ymd")
hat_all$Week <- as.Date(hat_all$Week, format="%Y-%m-%d")
hat_all <- as.data.frame(hat_all)
hat_all <- hat_all %>% mutate_if(is.character,parse_number)

# gather 사용
df1 <- gather(hat_all[, c("Week", "mu_z_naive", "mu_z_multi")],
              key = "Method", value = "mu_z", -Week)

df2 <- cbind(test_mon,single_beta)
```

```
## Warning in as.data.table.list(x, keep.rownames = keep.rownames, check.names
## = check.names, : Item 1 has 27 rows but longest item has 28; recycled with
## remainder.
```

```
df2 <- rename(df2, Week =V1)
df2$Week <- parse_date_time(df2$Week, "ymd")
df2$Week <- as.Date(df2$Week, format="%Y-%m-%d")
df2 <- as.data.frame(df2)
```

```
df2 <- df2 %>% mutate_if(is.character,parse_number)


g <- ggplot(df1) +
  geom_line(aes(x = Week, y = mu_z, color = Method, linetype = Method)) +
  geom_point(data=df2, aes(x = Week, y = single_beta, color = "single_beta")) +
  guides(linetype = "none") +
  scale_color_discrete(name = "Method")
g
```