BDA - Project

Atul Kumar

Contents

Introduction

Problem of Ranking appear when we want to know the underlying order of given data. For example given a list of movie on netflix and each user preference we want to know which movie comes in top 10. The chalenge here is that we don't have preference form each user for each movie. We know a small movie preferences for each user.

The problem is to figure out a universal order of movie ranking given a set of preferences provided by the user.

Dataset

For this experiment I used data from https://www.preflib.org/data/election/netflix/ I used .soc files 1 - 100 which all have 3 candidate elections. All .soc files are merged into one dataframe and movie names are mapped to index. The processing files are found here https://github.com/atulkum/BDA_project/blob/main /process_data.py

Overall there are 171 movies (indicated by N_options in the code). Also we have around 600 rows in the data each has a weight and 3 elections order of the movies given by the user.

```
1) movie name to id

id movie_name
-------------
1   Shrek (Full-screen)
2   The X-Files: Season 2
3   The Punisher
```

```
4   Spy Game
5   Lethal Weapon 4
6   Glengarry Glen Ross

2) User Ranking data

count rank1 rank2 rank3
-----------------------
263 2   1   3
249 1   2   3
78  1   3   2
46  2   3   1
17  3   1   2
11  3   2   1
```

Model

For this problem I used "Exploded Logit" https://en.wikipedia.org/wiki/Discrete_choice#exploded_logit

The stan code is taken from the this link and modified for weighted ranking data and used in netflix dataset. https://bnicenboim.github.io/2021/03/21/a-simple-way-to-model-rankings-with-stan/

Prior

The uniform Dirichlet distribution is used as prior, which indicate we are not certain that how sparse the probabilities are. The ranking score varibale theta is declared as a simplex so that all probability should sum to 1.

Stan model code

```
cat(readLines('./weighted_exploded.stan'), sep = '\n')
```

```
## functions {
##  real exploded_lpmf(int[] x, vector Theta){
##     real out = 0;
##     vector[num_elements(Theta)] thetar = Theta;
##     for(pos in x){
##       out += log(thetar[pos]) - log(sum(thetar));
##       thetar[pos] = 0;
##     }
##     return(out);
##  }
## }
## data{
##   int N_ranking; //total times the choices were ranked
##   int N_ranked; //total choices ranked
##   int N_options; //total options
##   int res[N_ranking, N_ranked];
##   int weights[N_ranking];
## }
## parameters {
##   simplex[N_options] Theta;
```

```
## }
## model {
##   target += dirichlet_lpdf(Theta| rep_vector(1, N_options));
##   for(r in 1:N_ranking){
##     for (w in 1:weights[r]){
##       target += exploded_lpmf(res[r]|Theta);
##     }
##   }
## }
```

inference from models

Following is the inference code for the model. 4 chains are used and each chain run for 2000 iteration of which 1000 is thrown awaya as warmup.

```
library(dplyr)
library(purrr)
library(ggplot2)
library(posterior)
library(cmdstanr)
library(bayesplot)

ldata <- list(res = as.matrix(netflix_data %>% select(starts_with("rank"))),
              weights = netflix_data$count,
              N_options = nrow(movie_names),
              N_ranked = ncol(netflix_data)-1,
              N_ranking = nrow(netflix_data))

m_expl <- cmdstan_model("./weighted_exploded.stan")

f_exploded <- m_expl$sample(
  data = ldata,
  seed = 42,
  parallel_chains = 4
)
saveRDS(f_exploded, file = "f_exploded.RDS")
```

Result

Following code the movie names are ranked by mean score of the draws and top 10 movies come out are

111 Lost in Translation
113 Men in Black II
115 Jurassic Park
73 Swordfish
94 Monster's Ball
95 Seven
39 The Silence of the Lambs
1 Shrek (Full-screen)
59 I
28 Say Anything

```
f_exploded <- readRDS("f_exploded.RDS")
movie_rankings = data.frame(
```

```
  by_mean = movie_names[order(theta_summary$mean, decreasing = TRUE)[1:10],"movie_name"],
  by_median = movie_names[order(theta_summary$median, decreasing = TRUE)[1:10],"movie_name"],
  by_mad = movie_names[order(theta_summary$mad, decreasing = TRUE)[1:10],"movie_name"],
  by_p95 = movie_names[order(theta_summary$q95, decreasing = TRUE)[1:10],"movie_name"]
)
movie_rankings
```

Simulation - Based calibration

To find the issues in the model I used "Simulation - Based calibration" The code used for this is given below.

Somehow the chi squre test for uniformality is giving 0 as p-value. I still need to investigate the model.

Reference https://mc-stan.org/docs/2_25/stan-users-guide/testing-uniformity.html

```
cat(readLines('./weighted_exploded_scb.stan'), sep = '\n')
```

```
## functions {
##  real exploded_lpmf(int[] x, vector Theta){
##     real out = 0;
##     vector[num_elements(Theta)] thetar = Theta;
##     for(pos in x){
##       out += log(thetar[pos]) - log(sum(thetar));
##       thetar[pos] = 0;
##     }
##     return(out);
##  }
##  int[] exploded_rng(int ranked, vector Theta){
##     int res[ranked] = rep_array(0,ranked);
##     vector[num_elements(Theta)] thetar = Theta;
##
##     for(i in 1:ranked){
##       thetar = thetar/sum(thetar);
##       res[i] = categorical_rng(thetar);
##       thetar[res[i]] = 0;
##     }
##     return (res);
##  }
## }
## data{
##   int N_ranking; //total times the choices were ranked
##   int N_ranked; //total choices ranked
##   int N_options; //total options
## }
## transformed data {
##   int res_sim[N_ranking, N_ranked];
##   vector[N_options] Theta_ = dirichlet_rng(rep_vector(1, N_options));
##
##   for (j in 1:N_ranking){
##     res_sim[j] = exploded_rng(N_ranked, Theta_);
##   }
## }
## parameters {
##   simplex[N_options] Theta;
```

```
## }
## model {
##   target += dirichlet_lpdf(Theta| rep_vector(1, N_options));
##   for(r in 1:N_ranking){
##     target += exploded_lpmf(res_sim[r]|Theta);
##   }
## }
##
## generated quantities {
##   int ranks_[N_options];
##   for (j in 1:N_options){
##     ranks_[j] = Theta[j] < Theta_[j];
##   }
## }
```

Simulation

```
df_all = NULL
M = 999
for (i in 1:M) {
  data_sim <- list(
                N_options = nrow(movie_names),
                N_ranked = ncol(netflix_data)-1,
                N_ranking = 10)


  m_expl_sim <- cmdstan_model("./weighted_exploded_scb.stan")

  f_exploded_sim <- m_expl_sim$sample(
    data = data_sim,
    parallel_chains = 4,
    refresh = 0
  )
  rank_stat_df <- as_draws_df(f_exploded_sim$draws("ranks_")) %>% select(starts_with("ranks_"))
  if (i == 1){
    df_all <- colSums(rank_stat_df)
  } else {
    df_all <- rbind(df_all, i=colSums(rank_stat_df))
  }
}
rownames(df_all)<-NULL
```

binning

```
J <- 50
n_choices <- 171
M_draw <- 4000
b <- matrix(0L, nrow = J, ncol = n_choices)
r <- matrix(as.numeric(unlist(df_all)),nrow=nrow(df_all)) #999 171
for (k in 1:n_choices){
  for (m in 1:M){
    idx <- 1 + floor(r[m, k] * J / (M_draw + 1))
    b[idx, k] <- 1 + b[idx, k]
```

```
  }
}
```

Chi-suqre test for uniformaliy

```
for (k in 1:n_choices){
  ej = (M_draw + 1)/J
  chi_square_value = floor(sum((b[, 1] - ej)^2/ej))
  print(pchisq(chi_square_value, df=J-1, lower.tail=FALSE))
}
```