Bakeoff Example

Paul Nguyen

2025-08-25

Bakeoff Example

To demonstrate one iteration of the bakeoff experiment, we will fit one instance of the cubic compositional model to one set of training/testing data. For this example, we'll run the simulation with fewer iterations (400) for faster compilation (normally 2000). The majority of the code is taken from study/study__bakeoff.R. You may need to run study/gen__bakeoff__data__splits.R to produce the training and testing splits.

```
set.seed(2)
library(tidyverse)
library(readxl)
library(rstan)
library(rlist)
library(tidyselect)
library(forcats)


study = "../../decathlon_simulation"
data_dir = "../../decathlon_simulation/data/"
script_dir = "../../decathlon_simulation/study/"
stan_dir = "../../decathlon_simulation/stan_mods/"

source(paste0(script_dir, "decathlon_funs.R"))
source(paste0(script_dir, "settings_bakeoff.R"))
load(paste0(data_dir, "test_split_list_general.RData"))
```

In the study, we use a high performance computing cluster. We will manually set a job id for this example, which would typically determine the model type and data splits. In our experiments, we typically ran one model at a time. In this example, we will compare one instance of each cubic model (baseline, simple, and compositional).

```
# setting up the experiment
args = 0
job_id = as.numeric(args[1]) + 1

type = as.character(settings$type[job_id])
comp = as.character(settings$comp[job_id])
prior = as.character(settings$prior[job_id])
pred_type = as.character(settings$pred_type[job_id])
iter = as.integer(settings$iter[job_id])
dec_events <- c("hundred_m", "long_jump", "shot_put",
                "high_jump", "four_hundred_m", "hurdles",
                "discus", "pole_vault", "javelin",
                "fifteen_hundred_m")
```

```r
# reading and standardizing data, organizing data split
online_data_filter <- read_csv(paste0(data_dir,"online_data_filter.csv"))
event_sums <- get_event_sums_df(online_data_filter)
dec_data_standard <- standardize_decathlon_data(online_data_filter,
                                                event_sums)
test_index = test_split_list[[iter]]
train_df <- dec_data_standard[-test_index,]
test_df <- dec_data_standard[test_index,]

athlete_id <- test_df$athlete_id
is_new_athlete <- as.integer(!(test_df$athlete_id %in% train_df$athlete_id))
age_vec <- test_df$age
```

Now, we can run each model.

```r
baseline_sim <- get_baseline_cubic_sim(age_vec = age_vec,
                                    athlete_id = athlete_id,
                                    is_new_athlete = is_new_athlete,
                                    decathlon_data = train_df,
                                    stan_dir = stan_dir,
                                    iter = 400,
                                    return_all = F)

simple_sim <- get_simple_cubic_sim(age_vec = age_vec,
                                athlete_id = athlete_id,
                                is_new_athlete = is_new_athlete,
                                decathlon_data = train_df,
                                event_sums = event_sums,
                                stan_dir = stan_dir,
                                iter = 400,
                                return_all = F)

comp_sim <- get_comp_cubic_sim(age_vec = age_vec,
                              athlete_id = athlete_id,
                              is_new_athlete = is_new_athlete,
                              decathlon_data = train_df,
                              event_sums = event_sums,
                              stan_dir = stan_dir,
                              iter = 400,
                              return_all = F)
```

Caculating SMSE's for each model. The SMSE of a model is its standardized mean squared error. This is calculated by taking the model's MSE and dividing it by the mean squared error if we had simply used the training mean as our test prediction for each observation. Lower SMSE's and MSE's are preferred.

```r
# baseline SMSE
pred_df_baseline <- data.frame(points = baseline_sim$post_pred)
baseline_mse = mean((pred_df_baseline[["points"]] - test_df[["points"]])^2,
                        na.rm = T)
denom = mean(((mean(train_df[["points"]]) - test_df[["points"]])^2),
                       na.rm = T)
baseline_smse = baseline_mse / denom
# simple and compositional SMSE

pred_df_simple <- simple_sim$sim_events %>%
```

```r
    group_by(athlete, age) %>%
    summarize(hundred_m = mean(hundred_m),
              long_jump = mean(long_jump),
              shot_put = mean(shot_put),
              high_jump = mean(high_jump),
              four_hundred_m = mean(four_hundred_m),
              hurdles = mean(hurdles),
              discus = mean(discus),
              pole_vault = mean(pole_vault),
              javelin = mean(javelin),
              fifteen_hundred_m = mean(fifteen_hundred_m),
              points = mean(calc_point, na.rm = T)
    )
pred_df_simple$athlete_id <- test_df$athlete_id

pred_df_comp <- comp_sim$sim_events %>%
    group_by(athlete, age) %>%
    summarize(hundred_m = mean(hundred_m),
              long_jump = mean(long_jump),
              shot_put = mean(shot_put),
              high_jump = mean(high_jump),
              four_hundred_m = mean(four_hundred_m),
              hurdles = mean(hurdles),
              discus = mean(discus),
              pole_vault = mean(pole_vault),
              javelin = mean(javelin),
              fifteen_hundred_m = mean(fifteen_hundred_m),
              points = mean(calc_point, na.rm = T)
    )
pred_df_comp$athlete_id <- test_df$athlete_id


mse_table <- data.frame(event = c(dec_events, "points"),
                        mse_simple = rep(NA, 11),
                        mse_comp = rep(NA, 11),
                        denom = rep(NA, 11),
                        smse_simple = rep(NA, 11),
                        smse_comp = rep(NA, 11),
                        iter = iter)
# unstandardizing test_df
for (event in dec_events) {
  event_mean <- event_sums %>%
    filter(event == !!event) %>%
    select(mean_score) %>%
    pull()
  event_sd <- event_sums %>%
    filter(event == !!event) %>%
    select(sd_score) %>%
    pull()
  test_df[event] <- (test_df[event] *event_sd ) + event_mean
  train_df[event] <- (train_df[event] *event_sd ) + event_mean

}
```

```r
for (i in 1:11) {
  event = mse_table$event[i]
  mse_table$mse_simple[i] = mean((pred_df_simple[[event]] - test_df[[event]])^2,
                                 na.rm = T)
  mse_table$mse_comp[i] = mean((pred_df_comp[[event]] - test_df[[event]])^2,
                               na.rm = T)
  mse_table$denom[i] = mean((mean(train_df[[event]]) - test_df[[event]])^2,
                            na.rm = T)
  mse_table$smse_simple[i] = mse_table$mse_simple[i] / mse_table$denom[i]
  mse_table$smse_comp[i] = mse_table$mse_comp[i] / mse_table$denom[i]
  mse_table$type = type
  mse_table$iter = iter
}

mse_table$smse_baseline <- c(rep(NA, 10), baseline_smse)
mse_table %>%
  select(event, smse_baseline, smse_simple, smse_comp) %>%
  mutate(smse_baseline = round(smse_baseline, 2),
         smse_simple = round(smse_simple, 2),
         smse_comp = round(smse_comp, 2))
```

```
##                event smse_baseline smse_simple smse_comp
## 1          hundred_m            NA        0.31      0.31
## 2          long_jump            NA        0.44      0.44
## 3           shot_put            NA        0.18      0.18
## 4          high_jump            NA        0.34      0.33
## 5     four_hundred_m            NA        0.36      0.36
## 6            hurdles            NA        0.32      0.32
## 7             discus            NA        0.27      0.27
## 8         pole_vault            NA        0.27      0.27
## 9            javelin            NA        0.27      0.27
## 10 fifteen_hundred_m            NA        0.44      0.44
## 11            points          0.24        0.24      0.24
```

Above, we present the results for each model for this one training and testing data split. In our experiments, we repeat the above process 10 times for each model combination.