VAEP Vignette

Robert Hickman

30/08/2020

First things first, we want the libraries we'll need to get the event data, convert it into SPADL format and extract features and labels, and then to train a model on this data. There's probably some performance hit using tidyverse for this job but on 60 games worth we're fine. Eventually this (and the repo as a whole) will probably be rewritten using data.table methods

```
library(tidyverse) #munging
library(Rteta) #convert to SPADL/vaep
library(StatsBombR) #get data
library(xgboost) #train xgboost model
```

Next we need to download our data. As with the public socceraction notebooks, we'll be using StatsBomb's free 2018 World Cup event data. This can easily be downloaded using StatsBomBR in a few lines of code. Downloading 64 matches worth of data will take a minute or two.

```
worldcup <- StatsBombR::FreeCompetitions() %>%
  dplyr::filter(competition_name == "FIFA World Cup")
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be responsible wi
```

```
worldcup_matches <- StatsBombR::FreeMatches(worldcup[,1:2])
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be responsible wi
```

```
worldcup_events <- StatsBombR::StatsBombFreeEvents(worldcup_matches)
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be responsible wi
```

Nest we want to convert this nested JSON into a nice 2D SPADL format using Rteta::. This is code is very alpha version so I'd espect some bugs. If you could file them with reproducible examples and maybe even fixes at the repo it would be much appreciated!

```
spadl <- worldcup_events %>%
  split(., f = .$match_id) %>%
  map_df(., Rteta::sb_convert_spadl)

head(spadl)
```

```
##   game_id period_id time_seconds    timestamp team_id home_team    team_name
## 1    7525         1        0.612 00:00:00.612     799     FALSE Saudi Arabia
## 2    7525         1        1.732 00:00:01.732     799     FALSE Saudi Arabia
## 3    7525         1        2.933 00:00:02.933     799     FALSE Saudi Arabia
## 4    7525         1        5.893 00:00:05.893     796      TRUE       Russia
## 5    7525         1        7.772 00:00:07.772     799     FALSE Saudi Arabia
## 6    7525         1        8.972 00:00:08.972     799     FALSE Saudi Arabia
##   player_id                 player_name action_id type_name bodypart_name
## 1      5196 Mohammad Ibrahim Al Sahlawi         1      pass          foot
```

```
## 2      5173      Abdullah Ibrahim Otayf         2   dribble          foot
## 3      5173      Abdullah Ibrahim Otayf         3      pass          foot
## 4      5175                  Ilya Kutepov       4      pass          foot
## 5      5178    Salman Mohammed Al Faraj         5      pass          foot
## 6      5187   Salem Mohammed Al Dawsari         6   dribble          foot
##   result_name start_x end_x start_y end_y end_z type_id result_id bodypart_id
## 1     success   52.06 75.00   33.57 32.71    NA       0         1           0
## 2     success   75.00 74.12   32.71 33.57    NA      21         1           0
## 3        fail   74.12 26.47   33.57 12.91    NA       0         0           0
## 4        fail   32.65 38.82   10.33  4.30    NA       0         0           0
## 5     success   38.82 25.59    4.30  4.30    NA       0         1           0
## 6     success   25.59 25.59    4.30  6.03    NA      21         1           0
```

Next we're going to extract features from this SPADL dataset. I haven't coded up a function to define which to take (I'm not sure I even want to), but if we want to use the socceraction trained xgboost model, each features needs to be in the correct column, so here's an object of column names that will do that. If you want to train your own model you can select features as you see fit!

```
features <- c(
  "type_id_a0", "type_pass_a0", "type_cross_a0", "type_throw_in_a0", "type_freekick_crossed_a0", "type_
  "type_corner_short_a0", "type_take_on_a0", "type_foul_a0", "type_tackle_a0", "type_interception_a0",
  "type_shot_freekick_a0", "type_keeper_save_a0", "type_keeper_claim_a0", "type_keeper_punch_a0", "type
  "type_non_action_a0", "type_dribble_a0", "type_goalkick_a0", "bodypart_foot_a0", "bodypart_head_a0",
  "result_fail_a0", "result_success_a0", "result_offside_a0", "result_owngoal_a0", "result_yellow_card_
  "goalscore_opponent", "goalscore_diff", "start_x_a0", "start_y_a0", "end_x_a0", "end_y_a0", "dx_a0",
  "dy_a0", "movement_a0", "start_dist_to_goal_a0", "start_angle_to_goal_a0", "end_dist_to_goal_a0", "en
)
```

So now we can run our SPADL data into the function to get the labels (when a goal is about to be scored/conceded) and features (where the action happens/ what type of action, etc. etc.). I split here by game but you could easily split by game-period, or even possession chain. This R-thonic notation will create some (small!) differences in feature output to the socceraction features, but they are negligible and I honestly prefer them this way.

If you want to train the model in R I left a few commented lines to split out some rows for training. In reality if you want to use this, I reckon you probably want to use some data you won't test on (e.g. if you're interested in League One players, using Premier League data, etc.) but each to their own.

For this vignette, I am using the model trained using socceraction in python which is attached to this package in data. I haven't quite got the xgboost APIs to match up between python and R as I'd like, but as with all changes, I think it's close enough for either to be usable and give actionable results. As the model is pre-trained, I don't need the training data.

```
#get the action features
vaep_features <- spadl %>%
  split(f = .$game_id) %>%
  map_df(., Rteta::vaep_get_features) %>%
  .[features]

#get the training objective labels - near future goals
vaep_labels <- spadl %>%
  split(f = .$game_id) %>%
  map_df(., Rteta::vaep_get_labels)

#in case you want to train the model in R too
#train_rows <- sample(seq(nrow(vaep_features)), round(nrow(vaep_features)/5))
#train_vaep_features <- vaep_features[train_rows,]
```

```
#train_vaep_labels <- vaep_labels[train_rows,]
```

(I still need to properly fix the package scoping of a model, but should be done in the next few days. You can find the data inside the source code for the package in the data folder/ on github here). VAEP works by training two xgboost model on the chance of any action leading to a goal or a concession in the next n (10) actions and then taking the difference between the value added of an action for each of these.

```
score_model <- xgboost::xgb.load("../data/vaep.scoremodel")
concede_model <- xgboost::xgb.load("../data/vaep.concedemodel")

#score_model <- xgboost::xgboost(data = as.matrix(vaep_features), label = as.matrix(vaep_labels["scores
#concede_model <- xgboost::xgboost(data = as.matrix(vaep_features), label = as.matrix(vaep_labels["conc
```

Once we have our model we can run it on some prepared data to predict the chance of any action leading to a goal in the near future, which we bind back to the original SPADL data frame

```
#prep - not strictly necessary but good practice
score_matrix <- xgboost::xgb.DMatrix(
    data = as.matrix(vaep_features),
    label = as.numeric(vaep_labels$scores)
)

concede_matrix <- xgboost::xgb.DMatrix(
    data = as.matrix(vaep_features),
    label = as.numeric(vaep_labels$concedes)
)

#bind predictions back to SPADL
spadl$scores <- predict(score_model, newdata = score_matrix)
spadl$concedes <- predict(concede_model, newdata = concede_matrix)
```

We then use the predictions of the model to calculate the value added of every action taken on the pitch (how much more likely an action makes a goal - how much more likely it makes conceeding).

```
#score actions
spadl <- spadl  %>%
  Rteta::vaep_get_scores("scores", "concedes")
```

As a quick demo, we can then use this to rank every player in the 2018 World Cup by how much value they added to their team's performance. To compare, you can see the results of the socceraction repo trained on the same data here.

```
#group by player and sum total VAEP
players <- spadl %>%
  dplyr::group_by(player_id, player_name) %>%
  dplyr::summarise(total_actions = n(),
            total_offense = sum(attack_score, na.rm = TRUE),
            total_defence = sum(defence_score, na.rm = TRUE),
            total_score = sum(vaep_value, na.rm = TRUE)
  ) %>%
  arrange(-total_score)

head(players)
```

```
## # A tibble: 6 x 6
## # Groups:   player_id [6]
##   player_id player_name    total_actions total_offense total_defence total_score
```

```
##       <int> <chr>                <int>         <dbl>         <dbl>       <dbl>
## 1      3009 Kylian Mbappé~         495          3.68        -0.157        3.53
## 2      3308 Kieran Trippi~         690          4.33        -0.845        3.48
## 3     20004 Paul Pogba             676          2.81         0.324        3.13
## 4      3621 Eden Hazard            691          3.26        -0.229        3.04
## 5      3244 John Stones            937          2.61         0.422        3.03
## 6      3089 Kevin De Bruy~         716          3.34        -0.314        3.03
```

We can also normalise this VAEP score by the total minutes played using Rteta::sb_get_mins_played and calculate the most impactful players, again seeing that it nicely lines up with what we get from running socceraction.

```
#get the time played by each player per game
mins <- worldcup_events %>%
  split(., f = .$match_id) %>%
  purrr::map_df(., Rteta::sb_getmins_played) %>%
  dplyr::group_by(player.id) %>%
  #convert to minutes
  dplyr::summarise(total_mins = sum(state_seconds, na.rm = TRUE) / 60) %>%
  #some players may appear in time played but have no spadl actions
  dplyr::filter(player.id %in% players$player_id)

players <- players %>%
  dplyr::left_join(mins, by = c("player_id" = "player.id")) %>%
  mutate(vaep_p90 = total_score / (total_mins / 90)) %>%
  #filter out players with less than 2 full games
  filter(total_mins > 180) %>%
  select(player_name, total_vaep = total_score, vaep_p90) %>%
  arrange(-vaep_p90)

head(players)
```

```
## # A tibble: 6 x 4
## # Groups:   player_id [6]
##   player_id player_name                          total_vaep vaep_p90
##       <int> <chr>                                     <dbl>    <dbl>
## 1      5186 Denis Cheryshev                           2.80    0.877
## 2      5574 Toni Kroos                                2.77    0.842
## 3      5473 Ahmed Musa                                1.90    0.757
## 4      3237 Sergio Leonel Agüero del Castillo         1.58    0.722
## 5      6196 Yerry Fernando Mina González              2.32    0.702
## 6      4319 Edinson Roberto Cavani Gómez              2.81    0.695
```

That's all for now. There's still plenty of bugs and things to optimise I'm sure but at least this is working (mostly) and hopefully gives those not so familiar with python and chancce to play with VAEP :)