ML Code

Zheng Pan

7/18/2021

Introduction

This is a sophisticated tutorial on Machine Learning using R programming language. The dependent value is continuous in this case.

Workflow

1. modeling
2. performance assessment
3. model optimization

code

prerequisites

- load required libraries

```
library('tidyverse')
library('mlr3')
library('mlr3verse')
library("mlr3viz")
library('reshape2')
library('GenSA')
```

If you get an error loading them, you should probably install or update some of required packages using :

```
install.packages("NAME OF PACKAGE")
```

- initialize file paths

```
# Change your path to your .csv
path_of_file <- '~/Desktop/ml_code/photocurrent.csv'
st1 <- as.data.frame(read.csv(file = path_of_file))
# Change your path for prediction set
path_for_prediction <- '~/desktop/predions.csv'
```

THe path above is path on my laptop, users shall change them according to their own laptops.

- Create a function that generates pearson heat map

```
pearson_heat <- function(df, corm = -1){
  defaultW <- getOption("warn")
  options(warn = -1)
  if(corm == -1){
    corm = cor(df)
  }
  options(warn = defaultW)
  res <- melt(corm) %>%
    ggplot(aes(Var1,Var2,fill = value)) +
    geom_tile(color = "black",alpha = 0.8) +
    theme(axis.text.x = element_text(angle = 90,
                                     hjust = 0.5, vjust = 0.5)) +
    scale_fill_gradient2() +
    theme(panel.border = element_blank(),
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          axis.line = element_line(colour = "black")) +
    xlab(NULL) +
    ylab(NULL)
  return(res)
}
```

- Create a function which selects feature by pearson correlation coefficient

```
fea_slc_bycor <- function(df, corr = 0.8){
  corm <- cor(df)
  name_of_features <- colnames(df)
  name_of_features_d <- name_of_features
  origin_fea_length <- length(name_of_features)
  for(q in 1:(origin_fea_length - 1)){
    fea_t <- name_of_features_d[q]
    other_fea_t <- name_of_features_d[(q+1):length(name_of_features_d)]
    de_fea <- names(corm[fea_t, other_fea_t][abs(corm[fea_t, other_fea_t]) >= corr])
    name_of_features <- name_of_features[!(name_of_features %in% de_fea)]
  }
  res <- df[, colnames(df) %in% name_of_features]
  return(res)
}
```

- import data
- prepare the prediction set

Code in this section generates a data.frame which is meant for prediction set. Note this, in this case, the number of descriptors which are suppuosed to change to generate a prediction set is three and in other cases the number might change and the code below shall change accordingly.
It's not the best code that generates prediction set(it might be slow for larger descriptor amount)

```
tsk_df1 <- lapply(unique(st1$X), function(s){
  df1 <- st1[st1$X == s, ][1, ][, c(-length(st1[st1$X == s, ][1, ]),
      -length(st1[st1$X == s, ][1, ]) + 1,
      -length(st1[st1$X == s, ][1, ]) + 2, -length(st1[st1$X == s, ][1, ])
      + 3)]
  df2 <- cbind(df1, Manufacture.Method = c(1,2,3))
  res1 <- lapply(1:nrow(df2), function(q){
    df_t <- df2[q, ]
    (cbind(df_t, Material.Ratio.PbI2.Cation = seq(0,2,0.1)))
  })
  df3 <- do.call(rbind, res1)
  res2 <- lapply(1:nrow(df3), function(q){
    # q = 1
    df_t <- df3[q, ]
    (cbind(df_t, Time.s. = seq(0,500,5)))
  })
  df4 <- do.call(rbind, res2)
  df4 <- as.data.frame(df4)
  df4$Current.A. <- NA
  df4
})
```

- feature engineering

```
features <- st[, -ncol(st)]
pearson_heat(features, corm = cor(features))
fea_new <- fea_slc_bycor(features, corr = 0.8)
pearson_heat(fea_new)
```

machine learning

- Task construction

```
tsk_df <- cbind(st_t[, colnames(fea_new)], current = st_t[, ncol(st_t)])
task <- TaskRegr$new(id = "task", backend = tsk_df, target = "current")
```

- pick up a learner

```
cl <- mlr_learners$keys()[35:44]
qq = 9
learner <- mlr_learners$get(cl[qq])
```

In this case, the learner name is regr.svm

- train and test

```
train_set <- sample(NN, 0.8 * NN)
test_set <- setdiff(seq_len(NN), train_set)
learner$train(task, row_ids = train_set)
prediction <- learner$predict(task, row_ids = test_set)
```

- performance assessment

```
measure <- msr("regr.rmse")
prediction$score(measure)
autoplot(prediction) +
  theme_bw()
df1 <- rbind(cbind(prediction$response, 1: length(prediction$response),
                   'response'), cbind(prediction$truth,
                    1:length(prediction$truth), 'truth'))
df1 <- as.data.frame(df1)
df1[, 1] <- as.numeric(df1[, 1])
df1[, 2] <- as.numeric(df1[, 2])
df1[df1[,3] == 'response', 1][df1[df1[,3] == 'response', 1] < 0] <- 0
colnames(df1) <- c('y', 'x','f')
ggplot(df1, aes(x = x, y = y, color = factor(f))) +
  geom_point() +
  labs(title = paste0('mse = ', prediction$score(measure), '\nmodel name: ', learner$id))
```

- output predictions

```r
learner$predict(task, row_ids = 361: 500)$response
ind_seq <- c(seq(NN + 1, nrow(st_t), 500000), nrow(st_t))
ind_seq_l <- lapply(1:(length(ind_seq) - 1), function(q){
  if(q != 1){
    return((ind_seq[q] + 1):ind_seq[q+1])
  }else{
    ind_seq[q]:ind_seq[q+1]
  }
})
pred <- lapply(1:length(ind_seq_l), function(q){
  prediction2 <- learner$predict(task, row_ids = ind_seq_l[[q]])
  data.frame(prediction2$response)
})
preds <- do.call(rbind, pred)
to_w <- st2[(NN+1): nrow(st2),]
to_w[, ncol(to_w)] <- preds
write_csv(to_w, file = paths)
```