Code for implementing the PSA and FPSA methods (examples included)

Dan Kluger

3/23/2022

In this file we write a function runPSA_FPSA that implements the PSA and FPSA methods described in Kluger et. al (2021). We provide some examples of implementation for the PSA and FPSA methods using the dataset processed in Kluger et. al (2021). The processed dataset is available here: https://zenodo.org/record/6376160.

Loading data and required packages

```
library(dplyr)
library(MASS)
library(randomForest)
load("CropTypeDatasetProcessed.RData") #Datasets from: https://zenodo.org/record/6376160
```

Defining function runPSA_FPSA

runPSA_FPSA takes an already trained classifier, the training data, the target data, and estimates of the distribution of the labels in specefic subsets of the target and returns classifications using both the PSA and FPSA methods. For reference it also returns predictions on the target dataset that do not involve making any prior or feature shift adjustments.

The input arguments for runPSA_FPSA are as follows:

1. trained_classifier: A classifier trained on the data in training_df.
2. training_df: A dataframe with the training data. It should include a column titled 'labels' as well as columns for the features. It may or may not include a column titled 'region'.
3. target_df: A dataframe with the target data. It should include the same features as training_df. It should include a column titled 'region'. It may or may not include a column titled 'labels'.
4. predict_formated: This should be a function that takes a trained classifier, a dataframe of the features, and return predictions of that classifier on that target dataframe. The function should return a list with the predicted classes titled 'class' as well as probability estimates for being in each class titled 'PostProbs'. Examples of functions that can be used as inputs here are lda_predict_formatted and rf_predict_formatted, defined in example 1. The rational for using this function is that in R, the format of inputs and outputs of the predict() function varies depending on the base classifier used. We use the predict_formated argument to prespecify how to make and extract predictions in the appropriate format for an already trained classifier.
5. label_dists_byRegion: Estimates of the distribution of the crop type in region based on aggregate-level government statistics (or some other prior source of information that does not use the labelled data in the target region). Each column should corresponds to a different crop type and each row should

correspond to a different region. Entries should be expressed in terms of probabilities, so each row should sum to 1.

The outputs of runPSA_FPSA is a named list with the following 3 vectors:

1. UAT_preds: These predictions involve no application of feature or prior shift adjustment methods. The predictions are made by naively applying the classifier to the target dataset. Here UAT stands for unadjusted transfer.
2. PSA_preds: These predictions are based on applying the Prior Shift Adjustment (PSA) method described in Section 2.1 of Kluger et. al (2021).
3. FPSA_preds: These predictions are based on applying the Feature and Prior Shift Adjustment (FPSA) method described in Section 2.2 of Kluger et. al (2021).

```
runPSA_FPSA <- function(trained_classifier,train_df,target_df,predict_formated,
                        label_dists_byRegion){

  #computing helpful quantities
  train_label_dist <- table(train_df$labels)/nrow(train_df)
  n_target <- nrow(target_df)
  K <- ncol(label_dists_byRegion)

  #Performing some checks and reformatting if necessary

  if(max(abs(rowSums(label_dists_byRegion)-1))>10^(-12)){
    print("Warning: row sums of label_dists_byRegion should each equal 1")
  }

  if(sum(names(train_label_dist)!=colnames(label_dists_byRegion))>0){
    print("Warning: train_label_dist and label_dists_byRegion should have have the same ordering and nu
    train_label_dist_old <- train_label_dist
    train_label_dist <- rep(NA,K)
    names(train_label_dist) <- colnames(label_dists_byRegion)
    for(j in 1:K){
      curr_idx <- which(names(train_label_dist_old)==colnames(label_dists_byRegion)[j])
      train_label_dist[j] <- train_label_dist_old[curr_idx]
    }
  }

  #Making baseline predictions (no prior or feature shift adjustments)
  naive_preds <- predict_formated(trained_classifier,target_df)
  UAT_preds <- naive_preds$class

  #Making a matrix where each row corresponds to a point in the target dataset
  #and each row gives the prior probabilities of being in each class.
  #This is done by considering the region the point is in
  #and the appropriate row of label_dists_byRegion.

  prior_dists_target <- matrix(data = NA,nrow = n_target,ncol = K)
  colnames(prior_dists_target) <- colnames(label_dists_byRegion)
  RegionsTarget <- target_df$region

  for(j in 1:K){
    for(reg in unique(RegionsTarget)){
      prior_dists_target[RegionsTarget==reg,j] <- label_dists_byRegion[reg,j]
```

```
  }
}

#Making a matrix where each row corresponds to a point in the target dataset
#and each row gives the training set probabilities of being in each class.
label_dists_trainMat <- matrix(train_label_dist,nrow = n_target,ncol = K,byrow = TRUE)
colnames(label_dists_trainMat) <- names(train_label_dist)

#Implementing Prior Shift Adjustment (PSA)
prob_ests_UAT <- naive_preds$PostProbs
prob_reweight_PSA <- prob_ests_UAT* prior_dists_target/label_dists_trainMat
PSA_preds <- names(train_label_dist)[max.col(prob_reweight_PSA)]

#Calculating quantities used in FPSA (Part 1: estimate b_k vectors )
X_train <- train_df
X_train$labels <- NULL
bk_mat <- matrix(NA,ncol = ncol(X_train),nrow = K) #b_k will be kth row of bk_mat
for(k in 1:K){
  #taking mean feature vector for the kth crop type
  bk_mat[k,] <- colMeans(X_train[train_df$labels==names(train_label_dist)[k],])
}

#Calculating quantities used in FPSA (Part 2: estimate a_k vectors )
RegionsTargetFactor <- as.factor(RegionsTarget)
unique_regionsTarget <- levels(RegionsTargetFactor)
n_reg <- length(unique_regionsTarget)
X_target <- target_df
X_target$region <- NULL
if(!is.null(X_target$labels)){
  X_target$labels <- NULL
}

ar_mat <- matrix(NA,ncol = ncol(X_target),nrow = n_reg) #a_r will be rth row of ar_mat
  for(r in 1:n_reg){
  reg <- unique_regionsTarget[r]
  #taking mean feature vector in region r
  barXr <- colMeans(X_target[RegionsTarget==unique_regionsTarget[r],])
  ar_temp <- barXr

    for(k in 1:K){
      ar_temp <- ar_temp-label_dists_byRegion[reg,k]*bk_mat[k,]
    }
    ar_mat[r,] <- ar_temp
  }

#Applying FPSA by subtracting vectors in a_r vectors from features in the target data
X_target_FSA <-  X_target-ar_mat[as.numeric(RegionsTargetFactor),]
FSA_preds <- predict_formated(trained_classifier,X_target_FSA)
prob_ests_FSA <- FSA_preds$PostProbs
prob_reweight_FPSA <- prob_ests_FSA* prior_dists_target/label_dists_trainMat
FPSA_preds <- names(train_label_dist)[max.col(prob_reweight_FPSA)]
```

```
 return(list(UAT_preds=UAT_preds,PSA_preds=PSA_preds,FPSA_preds=FPSA_preds))
}
```

Example 1: Using runPSA_FPSA to apply PSA and FPSA for LDA and Random Forest when training in Haute-Garonne and making predictions on the rest of Occitanie.

Step 0: Separate the data into a training and test set

```
#Training data (Haute-Garonne)
train_region <- "HauteGaronne"
df_train <- France_crop_type_df %>% filter(region==train_region)
df_train$region <- NULL

#Testing data (all of the remaining departments in Occitanie)
df_target <- France_crop_type_df %>% filter(region!=train_region)
```

Step 1: Train classifier as you normally would

```
set.seed(1)
#Fitting LDA
lda_fit <- lda(labels ~.,data=df_train)

#Fitting Random Forest
train_Y <- df_train$labels
train_X <- df_train
train_X$labels <- NULL
rf_fit <- randomForest(x = train_X,y = train_Y)
```

Step 2: Write a function that takes a trained classifier, and a dataframe of the features in some target dataset, and makes predictions for that dataframe. It should return the classifications and posterior probabilities for the predictions that are automatically output when applying the classifier.The PSA and FPSA methods can be used for any choice of base classifier as long as the base classification method produces estimates of the probability of an unlabeled data point being in each class. Here, we exhibit the use of the PSA and FPSA methods for both Linear Discriminant Analysis (LDA) and Random Forests (RF).

```
#Writing a function to make properly formatted predictions for
#classifiers trained with lda function
lda_predict_formatted <- function(inp_classifier,new_dat){
  new_dat2 <- new_dat
  if(!is.null(new_dat$region)){
    new_dat2$region <- NULL
  }
  if(!is.null(new_dat$region)){
    new_dat2$region <- NULL
  }
  lda_preds <- predict(inp_classifier,newdata = new_dat2)
  return(list(class=lda_preds$class,PostProbs=lda_preds$posterior))
}
```

```
#Writing a function to make properly formatted predictions for
#classifiers trained with randomForest package
rf_predict_formatted <- function(inp_classifier,new_dat){
  new_dat2 <- new_dat
  if(!is.null(new_dat$region)){
```

```
    new_dat2$region <- NULL
  }
  if(!is.null(new_dat$region)){
    new_dat2$region <- NULL
  }
  rf_class_preds <- predict(inp_classifier,newdata = new_dat2)
  rf_prob_preds <- predict(inp_classifier,newdata = new_dat2,type='prob')
  return(list(class=rf_class_preds,PostProbs=rf_prob_preds))
}
```

Step 3: Use the function runPSA_FPSA to implement PSA and FPSA for the LDA and Random Forest classifiers.

```
UAT_PSA_FPSA_predsLDA <- runPSA_FPSA(trained_classifier = lda_fit,
                     train_df = df_train, target_df = df_target,
                     predict_formated = lda_predict_formatted ,
                     label_dists_byRegion = OccitanieFrance_cropType_dist_fromGovStatistics)

UAT_PSA_FPSA_predsRF <- runPSA_FPSA(trained_classifier = rf_fit,
                     train_df = df_train, target_df = df_target,
                     predict_formated = rf_predict_formatted ,
                     label_dists_byRegion = OccitanieFrance_cropType_dist_fromGovStatistics)
```

We print the overall accuracies for the different methods in the table below

```
Accuracy_table <- matrix(NA,nrow=2,ncol = 3)
rownames(Accuracy_table) <- c("LDA","RF")
colnames(Accuracy_table) <- c("No adjustment","PSA method","FPSA method")

Y_target <- as.character(df_target$labels)
Accuracy_table[1,1] <- mean(UAT_PSA_FPSA_predsLDA$UAT_preds==Y_target)
Accuracy_table[1,2] <- mean(UAT_PSA_FPSA_predsLDA$PSA_preds==Y_target)
Accuracy_table[1,3] <- mean(UAT_PSA_FPSA_predsLDA$FPSA_preds==Y_target)

Accuracy_table[2,1] <- mean(UAT_PSA_FPSA_predsRF$UAT_preds==Y_target)
Accuracy_table[2,2] <- mean(UAT_PSA_FPSA_predsRF$PSA_preds==Y_target)
Accuracy_table[2,3] <- mean(UAT_PSA_FPSA_predsRF$FPSA_preds==Y_target)

print(round(Accuracy_table,5))
```

```
##     No adjustment PSA method FPSA method
## LDA       0.84833    0.88103     0.91238
## RF        0.78226    0.83423     0.84783
```

Example 2: Using runPSA_FPSA to apply PSA and FPSA for LDA and Random Forest when training in Aveyron and making predictions in Aude and Pyrénées-Orientales.

```
#Step 0:Separate the data into a training and test set

#Testing data (Aveyron)
df_train <- France_crop_type_df %>% filter(region=="Aveyron")
df_train$region <- NULL

#Testing data (Aude and Pyrénées-Orientales)
```

```
df_target <- France_crop_type_df %>% filter(region %in% c("Aude","PyreneesOrientales"))

#Step 1: Train classifier as you normally would

set.seed(2)
#Fitting LDA
lda_fit <- lda(labels ~.,data=df_train)

#Fitting Random Forest
train_Y <- df_train$labels
train_X <- df_train
train_X$labels <- NULL
rf_fit <- randomForest(x = train_X,y = train_Y)

#Step 2: Write a function to make properly formatted predictions for
#classifiers the classifier you trained
#this is already done in example 1 where we defined the functions
#lda_predict_formatted and rf_predict_formatted

#Step 3: Use the function runPSA_FPSA to implement PSA and FPSA
UAT_PSA_FPSA_predsLDA <- runPSA_FPSA(trained_classifier = lda_fit,
                    train_df = df_train, target_df = df_target,
                    predict_formated = lda_predict_formatted ,
                    label_dists_byRegion = OccitanieFrance_cropType_dist_fromGovStatistics)

UAT_PSA_FPSA_predsRF <- runPSA_FPSA(trained_classifier = rf_fit,
                    train_df = df_train, target_df = df_target,
                    predict_formated = rf_predict_formatted ,
                    label_dists_byRegion = OccitanieFrance_cropType_dist_fromGovStatistics)
```

We can print the overall accuracies for example 2 in the table below

```
Accuracy_table2 <- matrix(NA,nrow=2,ncol = 3)
rownames(Accuracy_table2) <- c("LDA","RF")
colnames(Accuracy_table2) <- c("No adjustment","PSA method","FPSA method")

Y_target <- as.character(df_target$labels)
Accuracy_table2[1,1] <- mean(UAT_PSA_FPSA_predsLDA$UAT_preds==Y_target)
Accuracy_table2[1,2] <- mean(UAT_PSA_FPSA_predsLDA$PSA_preds==Y_target)
Accuracy_table2[1,3] <- mean(UAT_PSA_FPSA_predsLDA$FPSA_preds==Y_target)

Accuracy_table2[2,1] <- mean(UAT_PSA_FPSA_predsRF$UAT_preds==Y_target)
Accuracy_table2[2,2] <- mean(UAT_PSA_FPSA_predsRF$PSA_preds==Y_target)
Accuracy_table2[2,3] <- mean(UAT_PSA_FPSA_predsRF$FPSA_preds==Y_target)

print(round(Accuracy_table2,5))
```

```
##     No adjustment PSA method FPSA method
## LDA       0.78912    0.79942     0.82653
## RF        0.47666    0.56005     0.65443
```

Example 3: Using runPSA_FPSA to apply PSA and FPSA for LDA and Random Forest when training in Siaya, Kenya when and making predictions in Bungoma and Busia.

Unfortunately, at the time of the writing of Kluger et. al (2021), no aggregate crop type statistics were available at the regional level from the year 2017 in Western Province, Kenya. Therefore, we could not use actual government statistics to produce estimates of the distribution of crop type labels in each region. Instead we will artificially generate some estimates that could plausibly obtained from "survey sampling" 250 points in each region. We do that by generating a random sample of 250 crop type labels in each region from the distribution of crop types observed in the labelled data.

```
#Artificially creating prior estimates for the distribution of the crop type
#distribution in implementing a synthetic sample survey of 250 labelled
#points per region
set.seed(3)

Kenya_region_crop_table <- table(Kenya_crop_type_df$region,Kenya_crop_type_df$labels)
Kenya_regions <- rownames(Kenya_region_crop_table)
crop_types <- colnames(Kenya_region_crop_table)

Kenya_cropType_dist_fromSyntheticSurvey <- matrix(NA,nrow = length(Kenya_regions),
                                                  ncol=length(crop_types))
rownames(Kenya_cropType_dist_fromSyntheticSurvey) <- Kenya_regions
colnames(Kenya_cropType_dist_fromSyntheticSurvey) <- crop_types


for(i in 1:length(Kenya_regions)){
  labelled_dat_prop <- Kenya_region_crop_table[i,]/sum(Kenya_region_crop_table[i,])
  synthetic_survey <- sample(x = crop_types,size = 250,
                             replace = TRUE,prob = labelled_dat_prop)

  for(k in 1:length(crop_types)){
    Kenya_cropType_dist_fromSyntheticSurvey[i,k] <-
      sum(synthetic_survey==crop_types[k])/250
  }

}
```

```
#Step 0:Separate the data into a training and test set
#Training data (Siaya)
df_train <- Kenya_crop_type_df %>% filter(region=="Siaya")
df_train$region <- NULL

#Testing data (Bungoma and Busia)
df_target <- Kenya_crop_type_df %>% filter(region %in% c("Bungoma","Busia"))

#Step 1: Train classifier as you normally would

set.seed(4)
#Fitting LDA
lda_fit <- lda(labels ~.,data=df_train)

#Fitting Random Forest
train_Y <- as.factor(df_train$labels)
train_X <- df_train
```

```
train_X$labels <- NULL
rf_fit <- randomForest(x = train_X,y = train_Y)

#Step 2: Write a function to make properly formatted predictions for
#classifiers the classifier you trained
#this is already done in example 1 where we defined the functions
#lda_predict_formatted and rf_predict_formatted

#Step 3: Use the function runPSA_FPSA to implement PSA and FPSA for both
#LDA and Random Forest
UAT_PSA_FPSA_predsLDA <- runPSA_FPSA(trained_classifier = lda_fit,
                    train_df = df_train, target_df = df_target,
                    predict_formated = lda_predict_formatted ,
                    label_dists_byRegion = Kenya_cropType_dist_fromSyntheticSurvey)

UAT_PSA_FPSA_predsRF <- runPSA_FPSA(trained_classifier = rf_fit,
                    train_df = df_train, target_df = df_target,
                    predict_formated = rf_predict_formatted ,
                    label_dists_byRegion = Kenya_cropType_dist_fromSyntheticSurvey)
```

We can print the overall accuracies for example 2 in the table below

```
Accuracy_table3 <- matrix(NA,nrow=2,ncol = 3)
rownames(Accuracy_table3) <- c("LDA","RF")
colnames(Accuracy_table3) <- c("No adjustment","PSA method","FPSA method")

Y_target <- as.character(df_target$labels)
Accuracy_table3[1,1] <- mean(UAT_PSA_FPSA_predsLDA$UAT_preds==Y_target)
Accuracy_table3[1,2] <- mean(UAT_PSA_FPSA_predsLDA$PSA_preds==Y_target)
Accuracy_table3[1,3] <- mean(UAT_PSA_FPSA_predsLDA$FPSA_preds==Y_target)

Accuracy_table3[2,1] <- mean(UAT_PSA_FPSA_predsRF$UAT_preds==Y_target)
Accuracy_table3[2,2] <- mean(UAT_PSA_FPSA_predsRF$PSA_preds==Y_target)
Accuracy_table3[2,3] <- mean(UAT_PSA_FPSA_predsRF$FPSA_preds==Y_target)

print(round(Accuracy_table3,5))
```

```
##     No adjustment PSA method FPSA method
## LDA       0.29340    0.41372     0.58825
## RF        0.48093    0.49678     0.54457
```