Cross-Validation

2023-07-11

```r
library("readr")
library("dplyr")
library("caret")
```

Preprocessing

Load data

```r
all_features <- read_csv('../extraction/all_features.csv', show_col_types=FALSE)
df <- read_csv('../extraction/clean_df.csv', show_col_types=FALSE)
```

Create dataframe

```r
df_cv <- all_features %>%
  inner_join(subset(df, select=c(cat_no, img, cor, sub, positive)),
             by=c("cat"="cat_no", "img"="img")) %>%
  mutate(cor = if_else(positive == 0, 1-cor, cor)) %>%
  mutate(brightness=brightness_score,
         contrast=contrast_score,
         edges=edge_score,
         saturation=saturation_score,
         visual_complexity=visual_complexity,
         symmetry=symmetry,
         colorfulness=colorfulness) %>%
  group_by(brightness, contrast, edges, saturation,
           visual_complexity, symmetry, colorfulness) %>%
  summarize(aes=mean(cor))
```

Train-test split

```r
set.seed(0)
index = sample(1:nrow(df_cv), 0.8*nrow(df_cv))

train <- df_cv[index,]
test <- df_cv[-index,]

# Check dimensions
dim(train)
```

```
## [1] 5476    8
```

```r
dim(test)
```

```
## [1] 1370    8
```

80-20 split for train-test.

Modeling

Create functions for evaluation metrics

```r
# Adjusted R2 and RMSE for linear model
eval_linear = function(model, df, predictions, target){
  resids = df[,target] - predictions
  resids2 = resids**2
  N = length(predictions)
  r2 = as.character(round(summary(model)$r.squared, 4))
  adj_r2 = as.character(round(summary(model)$adj.r.squared, 4))
  print(paste(adj_r2, "(Adjusted R2)"))
  print(paste(as.character(round(sqrt(sum(resids2)/N), 4)), "(RMSE)"))
}

# R2 and RMSE for ridge model
eval_ridge <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- round((1 - SSE / SST), 4)
  RMSE = round(sqrt(SSE/nrow(df)), 4)
  print(paste(R_square, "(R2)"))
  print(paste(RMSE, "(RMSE)"))
}
```

Linear model

```r
# Build linear model
lr <- lm(aes ~ brightness + contrast + edges + saturation +
           colorfulness + symmetry + visual_complexity, data=train)
summary(lr)
```

```
##
## Call:
## lm(formula = aes ~ brightness + contrast + edges + saturation +
##     colorfulness + symmetry + visual_complexity, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79809 -0.15451 -0.00479  0.14987  0.76410
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -2.881e-01  1.551e-02 -18.574  < 2e-16 ***
## brightness         2.623e-03  9.560e-05  27.434  < 2e-16 ***
## contrast           3.776e-03  1.961e-04  19.261  < 2e-16 ***
## edges              6.328e-03  3.364e-04  18.811  < 2e-16 ***
## saturation         1.600e-03  1.069e-04  14.969  < 2e-16 ***
## colorfulness       2.572e-03  2.135e-04  12.042  < 2e-16 ***
## symmetry          -9.350e-05  2.180e-04  -0.429    0.668
## visual_complexity -1.691e-04  2.167e-05  -7.805 7.09e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2106 on 5468 degrees of freedom
## Multiple R-squared:  0.5152, Adjusted R-squared:  0.5145
## F-statistic:   830 on 7 and 5468 DF,  p-value: < 2.2e-16
```

```
# Evaluate with train-test split
## Train
predictions <- predict(lr, newdata=train)
eval_linear(lr, train, predictions, target='aes')
```

```
## [1] "0.5145 (Adjusted R2)"
## [1] "0.2104 (RMSE)"
```

```
## Test
predictions <- predict(lr, newdata=test)
eval_linear(lr, test, predictions, target='aes')
```

```
## [1] "0.5145 (Adjusted R2)"
## [1] "0.2125 (RMSE)"
```

Simple linear model on only the train set produces near identical results to the regression on the full dataset used in the paper. Using a train and test set allows us to evaluate the RMSE for the predictive model. An RMSE of 0.21 means that on average, the model will make an error of 0.21 (where the full range of the target is [0, 1]) when predicting the aesthetic value of an image based solely on the selected features.

In addition, the R2 and RMSE are very similar for the train and test sets, indicating that no overfitting has occurred.

Ridge regression

```
# Build the ridge model and find optimal lambda

# Create vector with column names
cols_reg <- c('brightness', 'contrast', 'edges', 'saturation',
              'colorfulness', 'symmetry', 'visual_complexity', 'aes')
dummies <- dummyVars(aes ~ ., data=df_cv[,cols_reg])

## Train-test split
train_dummies <- predict(dummies, newdata=train[,cols_reg])
test_dummies <- predict(dummies, newdata=test[,cols_reg])

## Check dimensions
print(dim(train_dummies)); print(dim(test_dummies))
```

```
## [1] 5476    7

## [1] 1370    7
```

```
# Build ridge regression model
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-7
```

```
X <- as.matrix(train_dummies)
y_train <- train$aes
```

```r
X_test <- as.matrix(test_dummies)
y_test <- test$aes

# Define range of lambdas to test
lambdas <- 10^seq(2, -3, by=-.1)
ridge_reg <- glmnet(X, y_train, nlambda=25, alpha=0,
                    family='gaussian', lambda=lambdas)
# summary(ridge_reg)

# Find optimal lambda
cv_ridge <- cv.glmnet(X, y_train, alpha=0, lambda=lambdas)
optimal_lambda <- cv_ridge$lambda.min
optimal_lambda
```

```
## [1] 0.001
```

As the optimal value for lambda is close to 0, the punishing factor in the ridge regression will be very small and as a consequence, the ridge regression will not differ much from the OLS linear regression.

Evaluate ridge model

```r
# Train
predictions_train <- predict(ridge_reg, s = optimal_lambda, newx = X)
eval_ridge(y_train, predictions_train, train)
```

```
## [1] "0.5152 (R2)"
## [1] "0.2104 (RMSE)"
```

```r
# Test
predictions_test <- predict(ridge_reg, s = optimal_lambda, newx = X_test)
eval_ridge(y_test, predictions_test, test)
```

```
## [1] "0.5247 (R2)"
## [1] "0.2125 (RMSE)"
```

The ridge model has very similar values for R2 and RMSE as linear regression. This makes sense because the optimal lambda value was very low.

Just as in the linear model, the ridge model also shows that the train and test performance are very close. This means that the ridge model is not overfitting either.