Semi-Supervised Learning on MNIST using mlr3

Siyi Wei

#Google Summer of Code 2021.

First Dataset: PCA with MNIST

The idea of this project is rather simple. In image recognition dataset MNIST is one of the most famous benchmark tests. However, most of the machine learning models do not perform well on it since the difficulty to extract the patterns. We could apply feature engineering to improve such problems. PCA is one common used trick on reducing dimension. Since we have 784 dimensions for the dataset. We could use PCA to reduce dimension, resolve the sparsity of the dataset and apply regularization at the same time. In this project, we want to do a simple experiment. Since the data matrix is sparse. We want to apply PCA transformation on the dataset and compare the results with not apply PCA transformation.

Experiment: Compare estimations from PCA and naive model

- First we want to explore the performance of the naive classification algorithm on MNIST datasets. We could see even though the prediction is better than random guessing. But the classification errors are still large.
- We want to further explore the model performance using PCA, we first extract the 24 PCs from the datasets and use it as the features in the classification model. The model performance decreased. But we would like to know why by taking a closer look in the subgroup.
- For some of the sub-group (1,2,3) the model accuracy increases significantly. Some of them stay unchanged and some of them decrease significantly. For example, after the PCA transformation. The model is more likely to classify 9 as 4 or 2 as 6.
- We could potentially resolve this issue by increasing the number of PCs. However it could not perform significantly better than the original dataset since the tree model we have chosen.

```
#Create the Data Backend
sample_id = sample(c(1:10000), 0.8*10000)
dataset_mnist <- read_csv("./MNIST/train.csv")[1:10000,]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```
dataset_mnist$label = as.factor(dataset_mnist$label)

train <- dataset_mnist[sample_id,]
train_backend <- as_data_backend(train)
test <- dataset_mnist[-sample_id,]
test_backend <- as_data_backend(test)

#Create the Task
```

```
task_train_MNIST <- TaskClassif$new(id = "train", backend = train, target = "label")
task_test_MNIST <- TaskClassif$new(id = "test", backend = test, target = "label")

#Create the pipeline operations
scale_po = po("scale")
pca_po = mlr_pipeops$get("pca", param_vals = list(rank. = 24))
learner_po =  mlr_pipeops$get("learner", mlr_learners$get("classif.rpart"))

# Create the model
pca_learner = scale_po %>>% pca_po %>>% learner_po
learner = scale_po %>>% learner_po
```

```
# Train the naive classification tree model
learner$train(task_train_MNIST)
```

```
## $classif.rpart.output
## NULL
```

```
prediction <- learner$predict(task_test_MNIST)

# evaluate the performance
sum(prediction$classif.rpart.output$truth == prediction$classif.rpart.output$response)/2000
```

```
## [1] 0.6405
```

```
confusionMatrix(prediction$classif.rpart.output$truth, prediction$classif.rpart.output$response)$table
```

```
##            Reference
## Prediction   0   1   2   3   4   5   6   7   8   9
##          0 165   0   0   6   1  14   2   4   5   1
##          1   1 186   4   3   9   7   0   3  16   2
##          2  13  13 119   2   6   3   8  14  29   4
##          3  15   6  14  82   6  27  17   7  17  15
##          4   3   0  12   0 127   2  15   3  13  26
##          5  21   7  10   9   1  73   8  24  22  12
##          6  14   2  23   0  12   4 142   5  19   0
##          7   9   3   2   0  15   1   0 159   7   9
##          8   0   4   8   6   3  10   3   7 114   3
##          9   1   2   5   5  14   1   2  32   6 114
```

```
#Train the PCA classification tree model and evaluate the performance
pca_learner$train(task_train_MNIST)
```

```
## $classif.rpart.output
## NULL
```

```
prediction2 <- pca_learner$predict(task_test_MNIST)

sum(prediction2$classif.rpart.output$truth == prediction2$classif.rpart.output$response)/2000
```

```
## [1] 0.5965
```

```
confusionMatrix(prediction2$classif.rpart.output$truth, prediction2$classif.rpart.output$response)$table
```

```
##            Reference
## Prediction   0   1   2   3   4   5   6   7   8   9
##          0 123   0   1   1   2  17   8  40   3   3
##          1   0 183   5  18   0   4   9   0   6   6
```

```
##          2  18   0 105  31  10   1  23  14   7   2
##          3  13   3   4 111   0  34   7  19  14   1
##          4   2   0  16   1 109   8  16  18   5  26
##          5  17   0   0  26   9  81   6  24  20   4
##          6   1   1  23   0   4   2 183   7   0   0
##          7   0   2   7   1   0  21   2 141  10  21
##          8   3   0   3  33   7  26   2   5  76   3
##          9   2   0   4   1  10  32   8  26  18  81
```

Second Dataset: Explore COMPAS

For the Second part. We would like to explore the dataset COMPAS using mlr3 pipeline operations. I used the pre_processed COMPAS dataset from the fairness package. Here I quote the introduction of COMPAS datasets "COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is a popular commercial algorithm used by judges and parole officers for scoring criminal defendant's likelihood of reoffending (recidivism). It has been shown that the algorithm is biased in favor of white defendants, and against black inmates, based on a 2 year follow up study (i.e who actually committed crimes or violent crimes after 2 years). The pattern of mistakes, as measured by precision/sensitivity is notable." In the following study we want to explore whether the algorithm will give a biased prediction

- We could again use the classification Tree model to predict the Two Year Recidivism. There is a 33.68% Classification error and moreover, The False Positive is 186 and False Negative is 230. FN is slightly higher.

```
sample_id = sample(c(1:6172), 0.8*6172)
compas_train = compas[sample_id,]
compas_test = compas[-sample_id,]
rownames(compas_test) = seq_len(nrow(compas_test))

compas_train_backend = as_data_backend(compas_train)
compas_test_backend = as_data_backend(compas_test)

task_train_COMPAS <- TaskClassif$new(id = "train", backend = compas_train_backend, target = "Two_yr_Rec
task_test_COMPAS <- TaskClassif$new(id = "test", backend = compas_test_backend, target = "Two_yr_Recidi

learner = lrn("classif.rpart", cp = .01)
learner$train(task_train_COMPAS)
prediction <- learner$predict(task_test_COMPAS)

confusionMatrix(prediction$response, prediction$truth)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  516 217
##        yes 171 331
##
##                Accuracy : 0.6858
##                  95% CI : (0.6591, 0.7117)
##     No Information Rate : 0.5563
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.3581
##
```

```
##  Mcnemar's Test P-Value : 0.02234
##
##             Sensitivity : 0.7511
##             Specificity : 0.6040
##          Pos Pred Value : 0.7040
##          Neg Pred Value : 0.6594
##              Prevalence : 0.5563
##          Detection Rate : 0.4178
##    Detection Prevalence : 0.5935
##       Balanced Accuracy : 0.6776
##
##        'Positive' Class : no
##
```

- We want to ask the next question then. Among those wrong predictions. Will ethnicity or gender play a role?
- From gender. We could see we have a quite imbalanced dataset where male is more than female. This could potentially be a problem. Other than that, the prediction error for male is 34.33% and the prediction error for females is 30.96%. The prediction on females is more accurate than male.
- The False Positive rate for male is 32.92%. The False Negative rate is 35.72%. So the FN rate for male is higher than the FP rate. For females, the FP rate is 14.37% and the FN rate is 64.55%. We could see the algorithm is biased on gender.

```
#The male and female test datasets
male_id = as.numeric(rownames(compas_test[compas_test$Female == "Male",]))
female_id = as.numeric(rownames(compas_test[compas_test$Female == "Female",]))

prediction_male <- learner$predict(task_test_COMPAS, row_ids = male_id)
prediction_female <- learner$predict(task_test_COMPAS, row_ids = female_id)

prediction_male$confusion
```

```
##         truth
## response  no yes
##      no  383 169
##      yes 152 303
```

```
prediction_female$confusion
```

```
##         truth
## response  no yes
##      no  133  48
##      yes  19  28
```

- For ethnicity. The Afican-American has prediction error to be 34% and the other ethnicity has prediction error to be 33.3%. Which is quite close
- However, we could see there is a significant difference in False Positive Rate and False Negative rate. For African-American, the FP rate is 41.23% and the FN rate is 27.48. For the other race, the FP rate is 17% and the FN rate is 57.14%. So the algorithm is biased in ethnicity.

```
#The black and other race id.
black_id = as.numeric(rownames(compas_test[compas_test$ethnicity == "African_American",]))
other_id = as.numeric(rownames(compas_test[compas_test$ethnicity != "African_American",]))

prediction_black <- learner$predict(task_test_COMPAS, row_ids = black_id)
prediction_other <- learner$predict(task_test_COMPAS, row_ids = other_id)
```

```
prediction_black$confusion
```

```
##          truth
## response  no yes
##      no  183  91
##      yes 108 231
```

```
prediction_other$confusion
```

```
##          truth
## response  no yes
##      no  333 126
##      yes  63 100
```