Aprendizagem de Máquina - Lista 3

Gustavo Marques Zeferino

Luís Fernando Israel Assunção

5 de outubro de 2018

Objetivo

O objetivo do trabalho é criar classificadores para predizer o dígito correspondente de uma imagem. O conjunto de dados utilizado é o do site de competições de Machine Learning Kaggle e está disponível em https://www.kaggle.com/c/digit-recognizer.

Time no Kaggle

A competição não permitiu a criação de times com mais de um usuário. Talvez isto ocorra por ser uma competição de treino para iniciantes. De qualquer forma, as submissões foram feitas a partir da conta pessoal no site (nome do time: zeferino). A imagem da Figura 1 mostra a relação das submissões realizadas e a correspondente acurácia retornada pelo site.

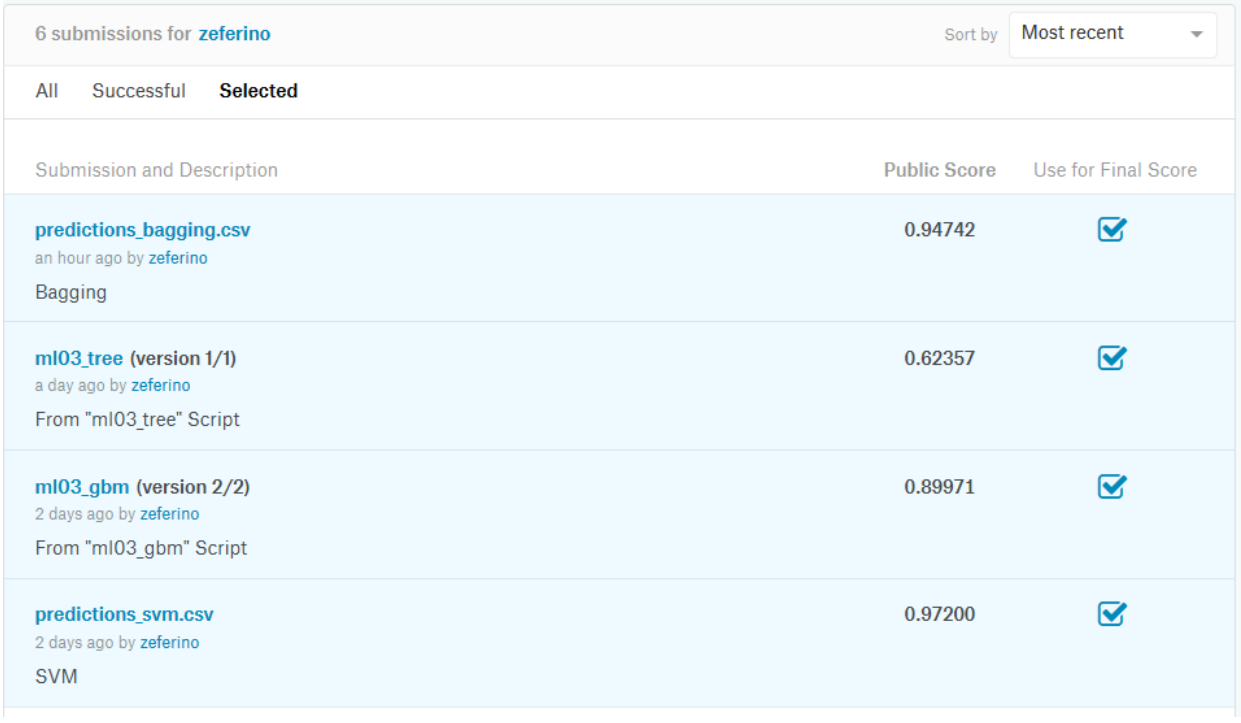

Figura 1: Submissões no site kaggle.com.

Descrição dos dados

O conjunto de dados é composto por 42.000 entradas em que cada uma representa uma imagem de dimensão 28 x 28 pixels (totalizando 784 covariáveis) e um rótulo com o dígito correspondente. O conjunto de validação disponibilizado pelo site contém 28.000 entradas.

Abaixo são exibidos os dígitos das 5 primeiras imagens do banco de dados na Figura 2.

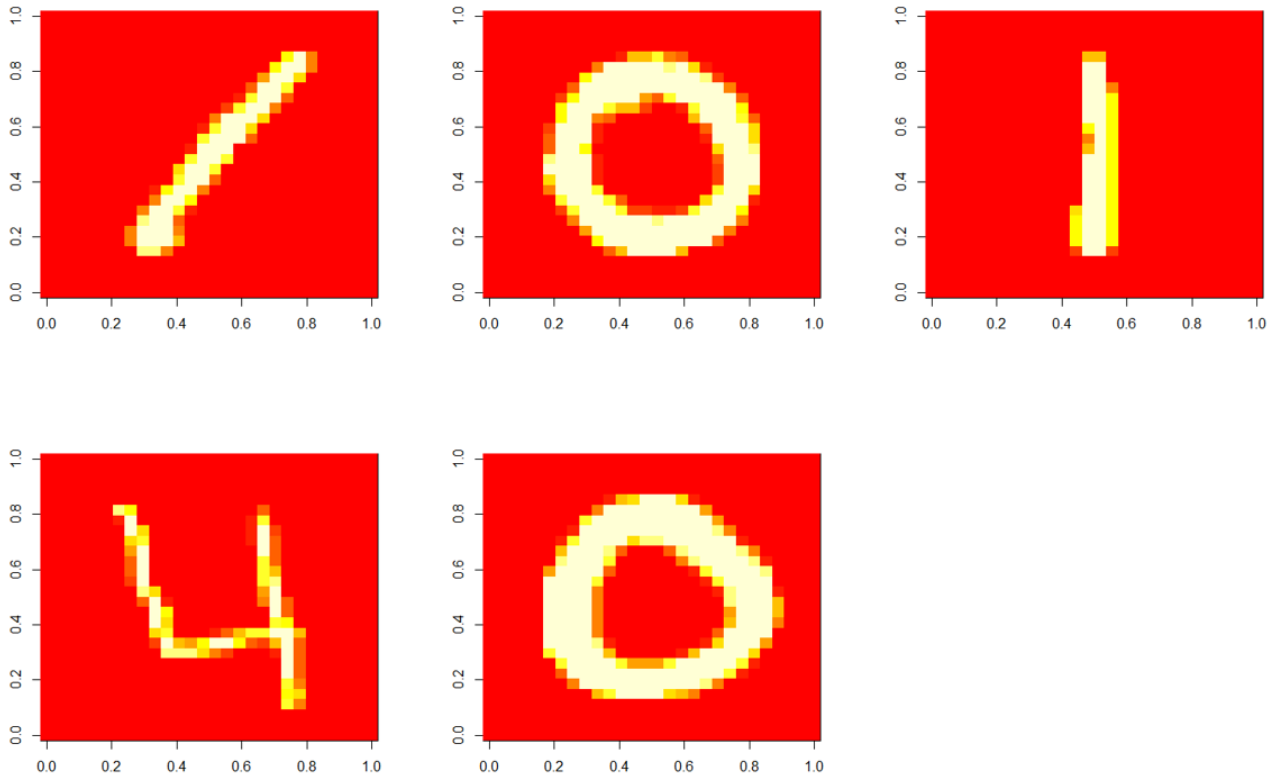

Figura 2: Imagem dos 5 primeiros dígitos da base de treino. Os dígitos correspondentes (de cima para baixo, da esquerda para direita): 1, 0, 1, 4 e 0.

Classificadores implementados

Com o objetivo de testar e ajustar a implementação de cada método, foi definido um conjunto de validação a partir do conjunto de treino disponível no site do desafio. Desta forma, 15% do conjunto de treino foi separado de forma aleatória para que fosse utilizado como o conjunto de dados para validação. O restante dos dados foi utilizado no treinamento e teste via validação cruzada.

Código para gerar a base de validação

```
set.seed(1)
train_test <- read.csv(file = 'train.csv', header = T)
train_test$label <- as.factor(train_test$label)
shuffle <- sample(nrow(train_test))
n_split <- 0.15 * nrow(train_test)
# Dados para treino dos algoritmos
my_train <- train_test[shuffle[1:n_split], ]
# Dados para validação
my_test <- train_test[-shuffle[1:n_split], ]
```

Bagging

Implementação

```
library(randomForest)

m <- dim(my_train)[2]
forest_model <- randomForest(label ~., data = my_train, mtry = m - 1)
forest_loss <- forest_model$err.rate[length(forest_model$err.rate)]

predicted_bagging <- predict(forest_model, newdata = my_test)
bagging_confusion <- confusionMatrix(predicted_bagging, my_test$label)
```

Árvores de Classificação

Implementação

```
library(tree)

tree_model <- tree(label ~., data = my_train)
cross_validate <- cv.tree(tree_model, FUN = prune.misclass)
best_size <- cross_validate$size[which.min(cross_validate$dev)]
tuned_tree <- prune.misclass(tree_model, best = best_size)

predicted_tree <- predict(tuned_tree, my_test, type = "class")
tree_confusion <- confusionMatrix(predicted_tree, my_test$label)
```

Boosting

Implementação

```
gb_fit <- gbm(label ~ ., distribution="multinomial", data=my_train,
              n.trees=300, shrinkage=0.1, cv.folds=5,
              verbose=TRUE, n.cores=4)

best_it <- gbm.perf(gb_fit, method='cv', plot.it = FALSE)

predicted_prob <- predict(gb_fit, n.trees = best_it, newdata = my_test, type=
'response')
predicted_gbm <- as.factor(apply(predicted_prob, 1, which.max) - 1)
gbm_confusion <- confusionMatrix(predicted_gbm, my_test$label)
```

SVM

Implementação

```
library(caret)

tc <- trainControl(method = "cv", number = 5, verboseIter = F, allowParallel
= T)
svm_fit <- train(label ~. , data= my_train, method = "svmRadial", trControl =
tc)

predicted_svm <- predict(svm_fit, newdata = my_test)
svm_confusion <- confusionMatrix(predicted_svm, my_test$label)
```

Comparação dos classificadores

Comparação via conjunto de validação "offline"

Dentre os métodos testados, o que obteve o melhor resultado foi o método SVM com um acurácia de 95,06% seguido pelo método Bagging com uma acurácia de 92,64%, Boosting com 88,18% e, por último, árvore de decisão com 63,01%.

Método	Acurácia	Intervalo de confiança 95%
Bagging	0,9264	(0,9237; 0,9291)
Tree	0,6301	(0,6250; 0,6351)
Boosting	0,8818	(0,8785; 0,8852)
SVM	0,9506	(0,9483; 0,9528)

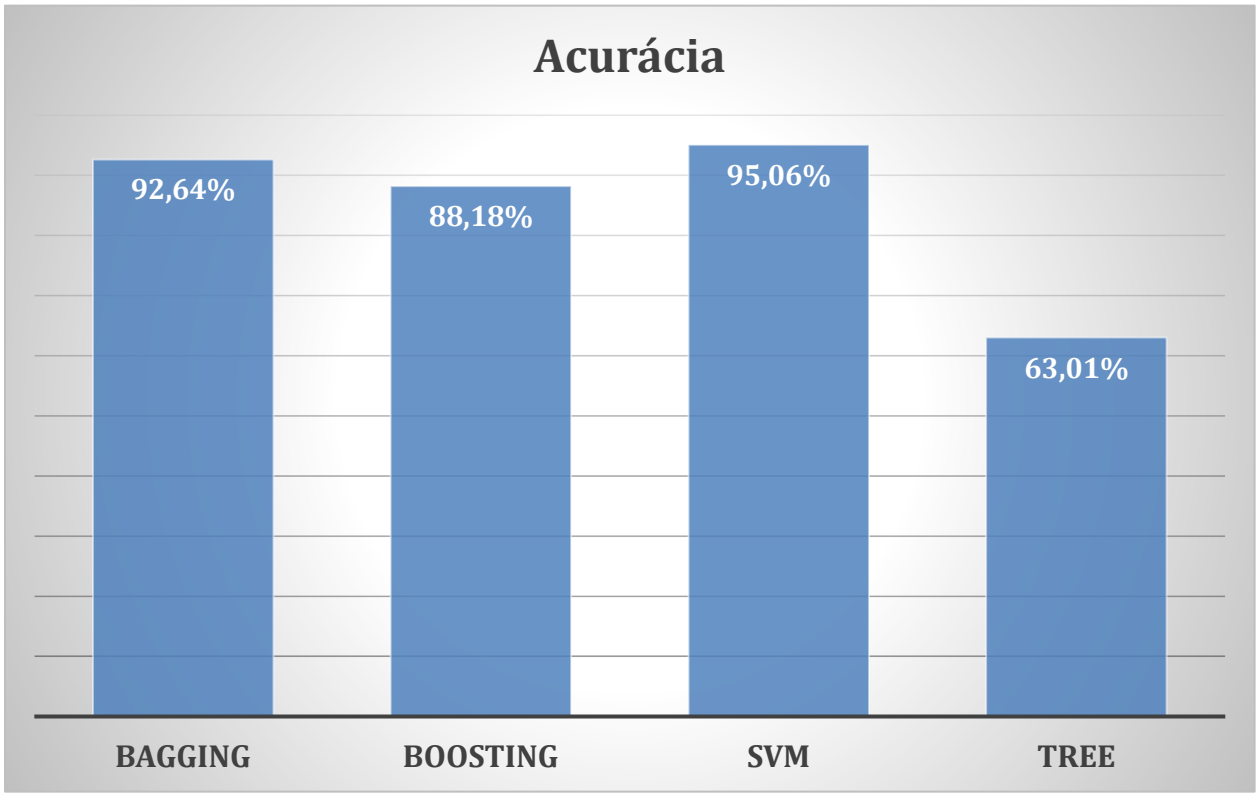

Figura 3: comparação da acurácia obtida no conjunto de validação gerado a partir da base de treino (corresponde a 15% da base de treino original).

Comparação via Kaggle

Para a comparação via conjunto de validação disponibilizado pelo Kaggle, todos os dados do conjunto de treino foram utilizados no treinamento dos algoritmos. Como o conjunto de validação não informa o rótulo correto, as predições foram submetidas no site e este informa qual a acurácia obtida.

No gráfico da Figura 4 é feito a comparação com a acurácia obtida na validação offline e na validação via site do desafio. Os valores são bem próximos nos dois métodos de validação e, como o site limite o número de submissões diárias, gerar uma base de validação é uma estratégia interessante para quem deseja fazer a comparação dos diversos métodos implementados de forma automatizada,

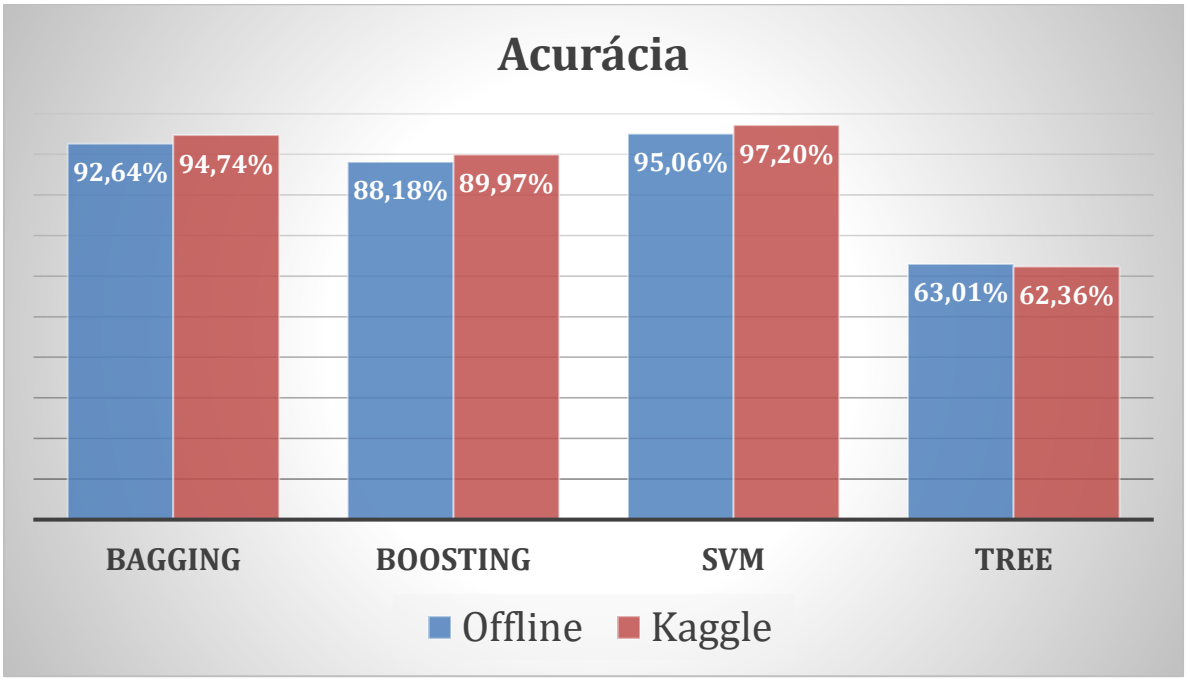

Figura 4; comparação da acurácia obtida na validação offline (com conjunto de validação gerado a partir da base de treino original) e a acurácia obtida no conjunto de validação fornecido pelo site do desafio.