kadaoui_alexandre_MNIST_et_Fashion_MNIST

Alexandre

22/12/2020

R Markdown

Dans le cadre de notre partiel, nous devons réaliser un total de 12 travaux retracant notre parcours et notre travail durant les 30 heures de cours.
Le travail à faire est le suivant :
- Une entête comportant un titre, un lien Github avec le ou les noms des auteurs.
- Une synthese de ce travail
- Un extrait commenté avec des parties de codes clé avec explication et commentaire.
- Une évalutation du travail avec nos 5 criteres.
- Une conclusion du travail

Definition des 5 critères de notations :

1) Effort de présentation :
2) Le knit est réalisable et bien présenté.
3) Explications simples et efficaces.
4) Le Code reproductible à d'autres DataFrame avec facilité.
5) Description des fonctions utilsés et du raisonnement.

"MNIST et Fashion MNIST (Illustration dans R)"

Travail réalisé par Kanlanfeyi Kabirou, Hounsinou Jordy.

https://github.com/kabirou7/PSBX (https://github.com/kabirou7/PSBX)

Synthese :

Le focus de ce tutoriel est la mise en lumière de deux bases de données afin d'illustrer les mécaniques de machine learning et de reconnaissance ainsi que leur utilité et possibles applications.
Les bases de données exploitées ici sont les bases MNIST (ou "Mixed National Institute of Standards and Technology", consistant en une bases de données d'images de chiffres écrits à la main) et Fashion MNIST (consistant en un jeu de données contenant 70 000 images en niveaux de gris, de 28x28 pixels, répartis sur 10 catégories différentes de vêtement).
La base FashionMNIST étant composée d'images d'objets complexes et non de simples chiffres, elle est plus complexe et donc plus utile afin d'étudier concrètement les application du machine learning dans un contexte se rapprochant du réel.
Il s'agira ici de créer puis entrainer deux modèles en parallèle, l'un sur MNIST et l'autre sur FashionMNIST afin de comparer leur pertinence et la précision de leurs capacités prédictives.

Extrait commenté du code :

Librairies nécessaires

```
library(readr)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```
library(class)

#Pour fractionner les données
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

#Lire les deux do{nnées: MNIST et fashion MNIST, CSV Manquants ? train.CSV et fashion.csv, naming imprécis + impossible de retravailler le code sans les bases de données utilisées, comment convertir bases de données de .gz à .CSV ???

```
mnist <- read_csv("train.csv")
fashion <- read_csv("fashion.csv")
```

```
mnist$label = factor(mnist$label)
fashion$label = factor(fashion$label)
```

```
mnist[,2:785] = mnist[,2:785]/255
fashion[,2:785] = fashion[,2:785]/255
```

Utilisation de DPLYR et de la fonction **sample_frac** afin de diviser l'échantillon de données, 80% sera utilisé pour entrainer le modèle, les 20% restants seront utilisés afin de tester laprécision de prédiction du modèle ainsi formé.

```
train_mnist <-  sample_frac(mnist, 0.8)
test_mnist <- anti_join(mnist, train_mnist)
train_fashion <-  sample_frac(fashion, 0.8)
test_fashion <- anti_join(fashion, train_fashion)
```

Construction des modèles

Random Forest

```
rf_MNIST <- randomForest(label ~ ., data = train_mnist, ntree = 10)
pred_MNIST1 <- predict(rf_MNIST, test_mnist)
rf_FASH <- randomForest(label ~ ., data = train_fashion, ntree = 10)
pred_FASH1 <- predict(rf_FASH, test_fashion)
```

Naive Bayes

```
bayes_MNIST <- randomForest(label ~ ., data = train_mnist)
pred_MNIST2 <- predict(bayes_MNIST, test_mnist)
bayes_FASH <- randomForest(label ~ ., data = train_fashion)
pred_FASH2 <- predict(bayes_FASH, test_fashion)
```

Dans les deux cas les fonctions utilisées à savoir **randomForest** et **predict** lors de la formation des modèles ne sont pas explicitées, de même que les méthodes utilisées à savoir “random forest” et “naive bayes”. Ces éléments bénéficieraient grandement à être expliqués plus en détails de manière concrète pour le lecteur.

Evalutation du travail :

1) Effort de présentation : La présentation du RMD ainsi que du PDF sont clairs, bien mis en pages, parfaitement lisible et esthétiquement très réussis.
2) Le knit est réalisable et bien présenté : Le knit, bien que très professionnel et esthétique dans l’exemple fourni sur github en PDF est malheureusement impossible à exécuter de mon côté dû à l’absence d“information sur les fichiers .csv”train.csv" et “fashion.csv”
3) Explications simples et efficaces : Les explication semblent assez confuses pour un lecteur débutant et certains concepts tels que les matrices de confusion, random forest ainsi que naive bayes auraient peut être pu être expliqués plus en détail. Le découpage des diverses étapes reste cependant très clair et lisible
4) Le Code reproductible à d’autres DataFrame avec facilité : Le code étant ici concentré sur ces deux bases clefs, l’adaptation à d’autres bases d’images nécessiterait quelques ajustements. Cependant ces ajustements seraient mineurs et celà grâce à la qualité du découpage du tutoriel.
5) Description des fonctions utilsés et du raisonnement : Les fonctions clefs utilisées ici sont **randomForest** ainsi que **predict** afin de construire deux modèles à partir des deux pools de données, **predict** afin de tester ces modèles et leur capacité de prédiction en les comparant à la réalité du pool de données de test.

Conclusion :

Il est malheureusement difficile d’émettre une réelle conclusion compte tenu des informations manquantes par rapports aux deux fichiers CSV ainsi que sur les fonctions **randomforest**. De plus, l’utilisation de fichiers CSV comme base de manipulation me semble peut être peu adapté à des usages concrets en condition, du moins pas sans ajouter une étape intermédiaire consistant à convertir les données des pixels composant l’image en fichier en format CSV. L’étude du package Keras et sa gestion des réseaux de neuronnes également présent sur le github aurait été une option intéressante cependant, dû à la densité du package et contraint par le temps, cette option sera étudiée à une date ultérieure.