MNIST et Fashion MNIST (Illustration dans R)

Groupe 12: Kanlanfeyi Kabirou, Hounssinou Jordy

11/16/2020

Sommaire

1. Introduction

De nos jours, les technologies de Machine Learning et de Deep Learning s'imposent dans tous les secteurs d'activités du fait de la numérisation des données. Ainsi, elles permettent à ces secteurs de réaliser des progrès spectaculaires grâce à l'exploitation de grands volumes de données.

Le but de ce document est de fournir une présentation succinte des bases de données MNIST et Fashion MNIST qui sont entre autres des bases de l'apprentissage de ces technologies novatrices. Il s'en suivra une application de certains algorithmes de Machine Learning afin de montrer leur utilité dans la prédiction de chiffres pour le cas de la base de données MNIST ou encore d'articles de mode en ce qui concerne la base de données Fashion.

2. Présentation des bases de données

L'acronyme MNIST (Modified ou Mixed National Institute of Standards and Technology), est une base de données de chiffres écrits à la main. C'est un jeu de données très utilisé en apprentissage automatique. Créée dans un premier temps pour répondre à un problème de reconnaissance de l'écriture manuscrite, la base MNIST est devenue un test standard grâce à son efficacité pour les algorithmes d'apprentissage. Elle regroupe 60,000 images d'apprentissage et 10,000 images de test, issues d'une base de données antérieure, appelée simplement NIST. Ce sont des images en noir et blanc, normalisées centrées de 28 pixels de côté. (source Wikipédia)

Figure 1: MNIST.

La Fashion MNIST est un jeu de données de qui contient elle aussi 70 000 images en niveaux de gris répartie sur 1 des 10 catégories. Dans ce cas, les images montrent des vêtements, d'articles de Zalando , en basse résolution (28 x 28 pixels). La répartition des données "Apprentissage-Test" étant similaire, cette base de données vise à remplacer le jeu de données MNIST (de chiffres écrit à la main) plus assez complexe dans une logique d'apprentissage automatique.

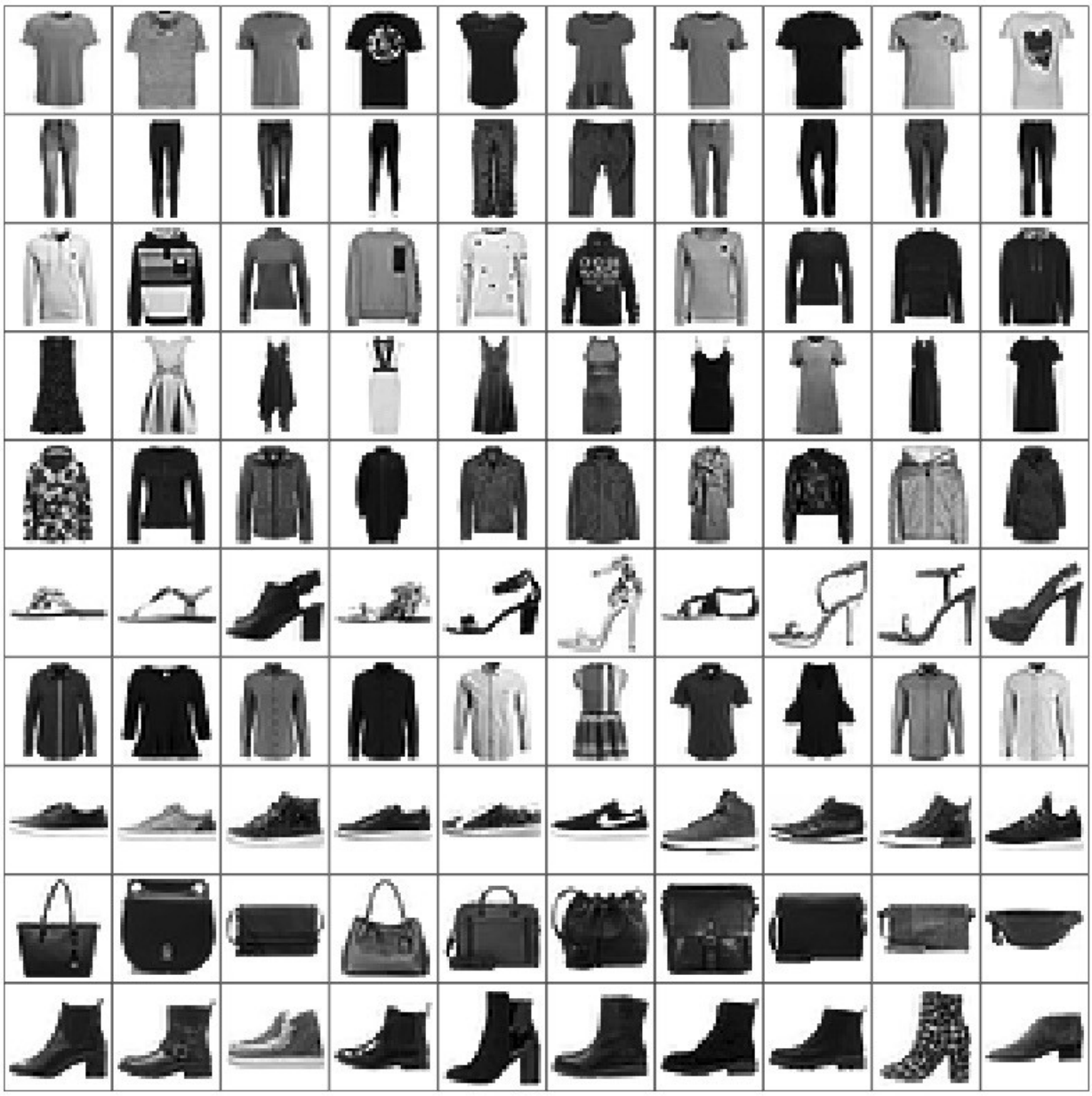

Figure 2: MNIST.

Le processus de reconnaissance des chiffres et des articles de mode est constitué de plusieurs étapes spécifiques et utilise différents algorithmes de Machine Learning ou de Deep Learning. Nous vous proposons dans les prochaines lignes de monnayer cette procédure et de vous en expliquer les étapes.

3.Algorithmes choisis

4.Exemple de code R

Dans cette partie nous allons utiliser des algorithmes de Machine Learning pour entrainer nos images afin de pouvoir en faire une prédiction des images que nous avons. Cette partie sera donc subdivisée en plusieurs section

```
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(xgboost)
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:xgboost':
##
##     slice

## 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
```

```
#Librairies Installée
#caret pour la matrice de confusion
#naiveBayes et randomForest : Algo``

#Lire les deux données: MNIST et fashion MNIST
mnist <- read_csv("train.csv")
```

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

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

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

Encodage de la colonne label sous forme de catégorie avec la fonction **factor**

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

Dans cette partie nous allons afficher les contenues des deux données avec la fonction head

```
#Visualisation de la structure des données en affichant les premières lignes
head(mnist[1:10,1:10])
```

```
## # A tibble: 6 x 10
##   label pixel0 pixel1 pixel2 pixel3 pixel4 pixel5 pixel6 pixel7 pixel8
##   <fct>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 1          0      0      0      0      0      0      0      0      0
## 2 0          0      0      0      0      0      0      0      0      0
## 3 1          0      0      0      0      0      0      0      0      0
## 4 4          0      0      0      0      0      0      0      0      0
## 5 0          0      0      0      0      0      0      0      0      0
## 6 0          0      0      0      0      0      0      0      0      0
```

```
head(fashion[1:10,1:10])
```

```
## # A tibble: 6 x 10
##   label pixel1 pixel2 pixel3 pixel4 pixel5 pixel6 pixel7 pixel8 pixel9
```

```
##   <fct>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 2          0      0      0      0      0      0      0      0      0
## 2 9          0      0      0      0      0      0      0      0      0
## 3 6          0      0      0      0      0      0      0      5      0
## 4 0          0      0      0      1      2      0      0      0      0
## 5 3          0      0      0      0      0      0      0      0      0
## 6 4          0      0      0      5      4      5      5      3      5
```

Nous allons alors scinder nos données en 2 parties: -une pour l'entrainement des données afin de construire des modèles prédictifs avec des algorithmes de machine Learning -une autre partie pour tester notre modèle construit

Nous allons utiliser la fonction sample!frac qui est fournie par la librairie *dplyr* Dans notre cas nous allons choisir un ratio de 80% pour l'entrainement et le reste pour le test

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

```
## Joining, by = c("label", "pixel0", "pixel1", "pixel2", "pixel3", "pixel4", "pixel5", "pixel6", "pixel
```

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

```
## Joining, by = c("label", "pixel1", "pixel2", "pixel3", "pixel4", "pixel5", "pixel6", "pixel7", "pixel
```

Construction de nos modèles:
Notons que nous construirons deux fois le même algorithme pour les deux jeux de données différents

Nous aimerons ajouter que pour la construction de nos modèles, nous avons minimisé l'ajout de paramètres au strict nécéssaire pour faciliter la durée de traitement.

###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)
```

###Utilisation de la Matrice de confusion pour évaluer notre model construit

```
cm_rf1 <- confusionMatrix(pred_MNIST1, test_mnist$label)
cm_rf2 <- confusionMatrix(pred_FASH1, test_fashion$label)
cm_nb1 <- confusionMatrix(pred_MNIST2, test_mnist$label)
cm_nb2 <- confusionMatrix(pred_FASH2, test_fashion$label)
```

###Affichage des résultats
Après les test de nos modèles sur les deux bases de données, nous nous retrouvons avec les résultats suivants

```
#Nous créons une matrice 2x2
valeurs <- matrix(c(cm_nb1$overall["Accuracy"],cm_nb2$overall["Accuracy"],cm_rf1$overall["Accuracy"],cm
colnames(valeurs)<- c("Naive Bayes", "Random Forest")
rownames(valeurs)<- c("MNIST", "Fashion MNIST")
tableau <- as.table(valeurs)
print(tableau)
```

```
##               Naive Bayes Random Forest
## MNIST           0.9686905     0.9386905
## Fashion MNIST   0.8818083     0.8563683
```

###Tableau récapitulatif

#Références

https://www.kaggle.com/c/digit-recognizer
https://www.kaggle.com/zalando-research/fashionmnist
https://www.kaggle.com/arathee2/random-forest-vs-xgboost-vs-deep-neural-network
https://thinkr.fr/premiers-pas-en-machine-learning-avec-r-volume-4-random-forest
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/factor
https://www.rdocumentation.org/packages/e1071/versions/1.7-3/topics/naiveBayes