Présentation du rapport

Diaby DRAME

Réseaux de neurones

Janvier 2023

Plan de l'exposé

Présentation

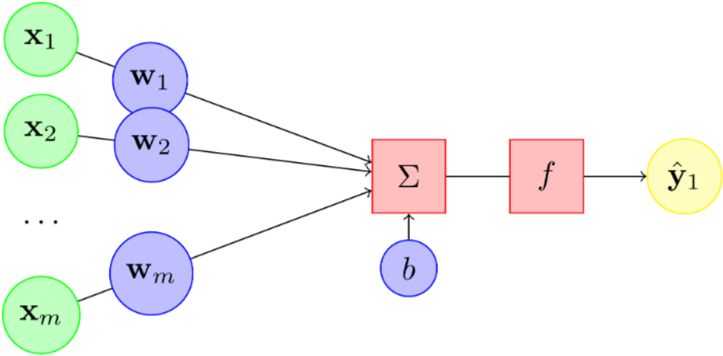


Figure – Un neurone formel.

$$\sum_{i=1}^{m} wixi + b$$

fonction identité $f(x) = x$
fonction sigmoïde $f(x) = \frac{1}{1+e^{-x}}$
fonction tangentielle $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$
fonction RELU $f(x) = x^{+} = max(0, x)$

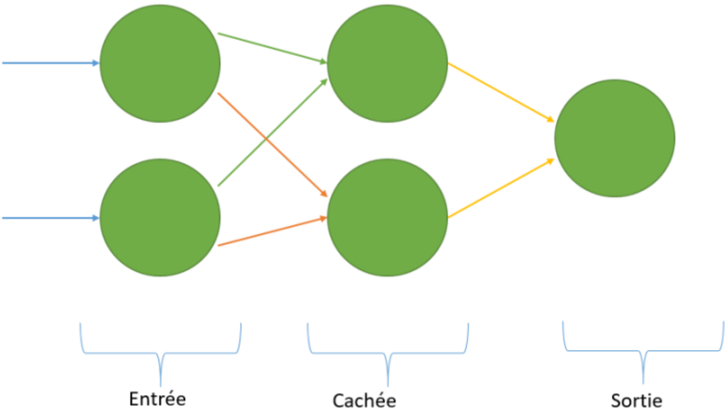


Figure – Perceptron multi-couches.

Apprentissage

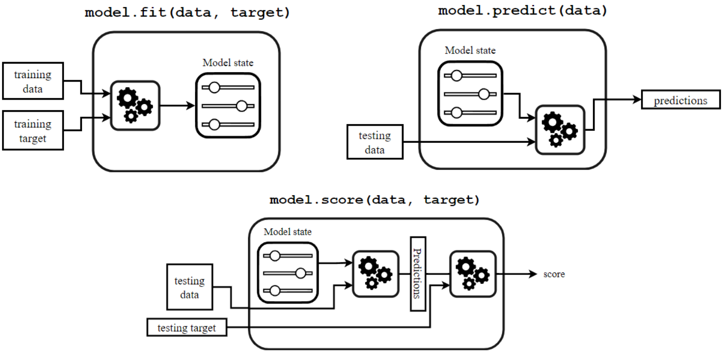
- **Minimisation de la fonction coût**
- **Descente du gradient**
- **Rétro-propagation**

Exemple d'utilisation

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long	gender
0	1	11.8	6.1	1	0	1	1	Male
1	0	14.0	5.4	0	0	1	0	Female
2	0	11.8	6.3	1	1	1	1	Male
3	0	14.4	6.1	0	1	1	1	Male
4	1	13.5	5.9	0	0	0	0	Female

Figure – cinq premières lignes du jeu de donnée.

Sciki-learn


model.fit(data, target)
training data
training target
Model state
model.predict(data)
Model state
testing data
predictions
model.score(data, target)
Model state
testing data
testing target
Predictions
score

```python
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
mlp_modele = MLPClassifier(random_state=0)

parametre = {'hidden_layer_sizes': [(10,10), (50,), (100,), (200,), (300,)],
                'activation': ['identity', 'logistic', 'tanh', 'relu'],
                'solver': ['lbfggs','sgd', 'adam']}

grid = GridSearchCV(mlp_modele, parametre)

grid.fit(data_train, target_train)

print(f"Les meilleurs paramètres sont : {grid.best_params_}")
```

```
Les meilleurs paramètres sont : {'activation': 'logistic', 'hidden_layer_sizes': (200,), 'solver': 'sgd'}
```

```python
from sklearn.ensemble import RandomForestClassifier

modele = RandomForestClassifier(random_state=0)
parametre = {'n_estimators' : [10,20,30,40,50,60,70,80,90,100],
             'max_depth' : [1,2,3,4,5,6,7,8,9,10]}
grid = GridSearchCV(modele,parametre)
grid.fit(data_train,target_train)

print(f"Les meilleurs paramètres sont : {grid.best_params_}")
```

```
Les meilleurs paramètres sont : {'max_depth': 4, 'n_estimators': 90}
```

Modèles	Score (train)	Score (test)
PMC	0.9654285714285714	0.9700199866755497
RF	0.9771428571428571	0.9706862091938707

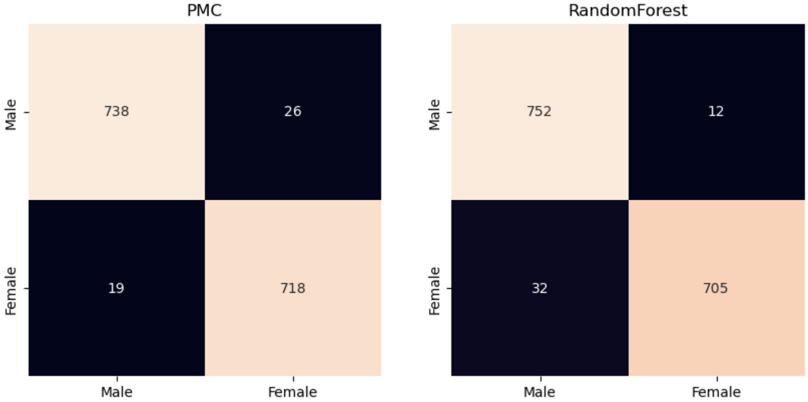


Figure – Matrice de confusion.

Keras

```python
from keras import layers, models

modele = models.Sequential()
modele.add(layers.Dense (200,activation='sigmoid',input_shape=(data_train.shape[1],)))
modele.add(layers.Dropout (0.3))
modele.add(layers.Dense(1,activation ='sigmoid'))
```

Figure – Confuguration du réseau.

```python
modele.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
modele.fit(data_train,target_train ,batch_size=40, epochs=50)


modele.evaluate(data_test , target_test)
```

```
Epoch 1/50
59/59 [==============================] - 1s 3ms/step - loss: 0.0889 - accuracy: 0.9586
Epoch 2/50
59/59 [==============================] - 0s 3ms/step - loss: 0.0852 - accuracy: 0.9603
Epoch 3/50
59/59 [==============================] - 0s 3ms/step - loss: 0.0860 - accuracy: 0.9571
Epoch 4/50
59/59 [==============================] - 0s 3ms/step - loss: 0.0827 - accuracy: 0.9620
Epoch 5/50
59/59 [==============================] - 0s 3ms/step - loss: 0.0844 - accuracy: 0.9600
Epoch 6/50
59/59 [==============================] - 0s 3ms/step - loss: 0.0841 - accuracy: 0.9626
Epoch 7/50
59/59 [==============================] - 0s 3ms/step - loss: 0.0866 - accuracy: 0.9611
```

Score : 0.9694

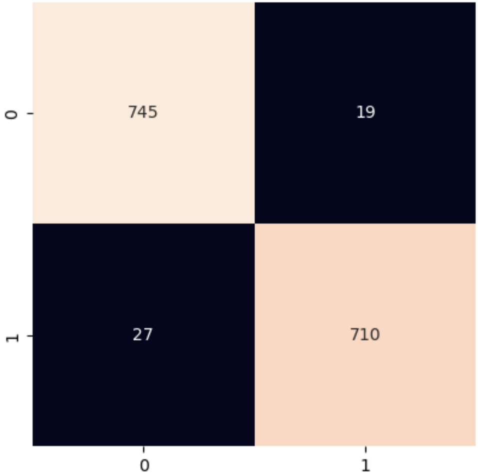


Figure – Matrice de confusion.

- Non supervisé
- Réduction de dimension
- Propriété topologique
- Visualisation

```
```{r}
library(kohonen)

data = read.csv("gender.csv",stringsAsFactors = F)

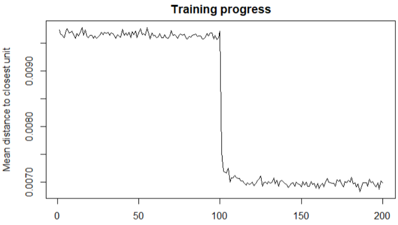
data = scale(data[,-8]) #On enlève l'étique sur nos données tout en normalisant les valeurs

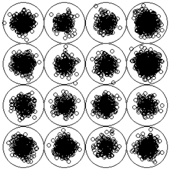
set.seed(0) # fixé la graine

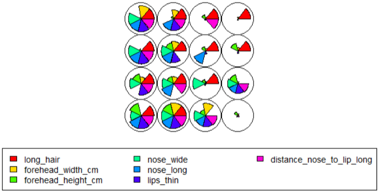
grille = somgrid(4,4,topo="rectangular")

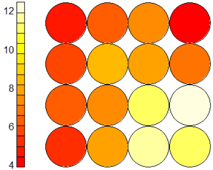
map = som(data,grid=g,rlen = 200) # nombre d'itération 200

plot(map,type = 'change')
plot(map,type='mapping')
plot(map, type = "codes", main = "Codes Plot", palette.name = rainbow)
plot(map, type = "dist.neighbours", main = "Distance voisinage")
```
```

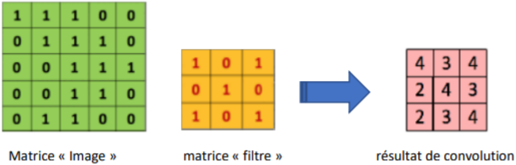

Training progress
Mean distance to closest unit
0.0070
0.0080
0.0090
0
50
100
150
200



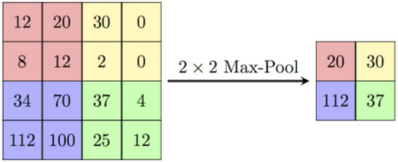

long_hair
forehead_width_cm
forehead_height_cm
nose_wide
nose_long
lips_thin
distance_nose_to_lip_long


12
10
8
6
4

- Couche de convolution



- Fonction Relu
- Couche de Pooling



Mnist

- 60.000 train et 10.000 test
- 28x28 pixels représentant un chiffre manuscrit
- 784 variables explicatives et une variable étiquette.

```{r}
library(keras)
modele <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = 'relu', input_shape =
c(28,28,1)) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = 'relu') %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_dropout(0.2) %>%
  layer_flatten() %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 10, activation = 'softmax')
```

```{r}
modele %>% compile(loss = loss_categorical_crossentropy,
                   optimizer = optimizer_adadelta(),
   metrics = c('accuracy'))

modele %>% fit(x_train,y_train,batch_size = 70,epochs = 20)
```

```
Epoch 1/20
858/858 [==============================] - 30s 33ms/step - loss: 0.9357 - accuracy: 0.6997
Epoch 2/20
858/858 [==============================] - 24s 28ms/step - loss: 0.9172 - accuracy: 0.7101
Epoch 3/20
858/858 [==============================] - 22s 26ms/step - loss: 0.9074 - accuracy: 0.7095
Epoch 4/20
858/858 [==============================] - 27s 31ms/step - loss: 0.8879 - accuracy: 0.7175
Epoch 5/20
858/858 [==============================] - 24s 28ms/step - loss: 0.8709 - accuracy: 0.7240
Epoch 6/20
858/858 [==============================] - 21s 25ms/step - loss: 0.8621 - accuracy: 0.7234
Epoch 7/20
858/858 [==============================] - 25s 30ms/step - loss: 0.8393 - accuracy: 0.7349
Epoch 8/20
858/858 [==============================] - 24s 28ms/step - loss: 0.8264 - accuracy: 0.7368
Epoch 9/20
```

```{r}
modele%>% evaluate(x_test, y_test)
```

```
     loss  accuracy
0.4409341 0.8925000
```

MERCI