Introduction : Apprentissage automatique

La classification d'images est une tâche usuelle en traitement d'image satellite. Que se soit en version non-supervisée pour simplifier le contenu des images ou bien en superviser pour aller vers l'automatisation de la classification des pixels des images dans des classes d'occupation du sol, de type de surface, etc. la mise en place de procédure d'apprentissage automatique (machine learning) est de plus en plus courante. Ces fonctionnalités de classification automatique sont disponibles dans les outils de traitement d'image. L'objectif est ici de vous rendre plus autonomes vis-à-vis de ces solutions logicielles propriétaires : à la fois pour être en mesure de développer des analyses sans avoir à payer la licence onéreuse de ces outils mais également en s'affranchissant des limitations imposées par des choix propres à ces outils.
En utilisant la programmation à l'aide d'outils libres, vous vous donnez accès à une panoplie de méthode d'apprentissage (et de validation) qui vous permettront de décider en connaissance des meilleurs méthodes d'apprentissage à utiliser pour vos données.

En général, un problème d'apprentissage considère un ensemble de n échantillons de données et essaie ensuite de prédire les propriétés de données inconnues. Si chaque échantillon est plus qu'un seul nombre et, par exemple, une entrée multidimensionnelle (données multivariées), on dit qu'il a plusieurs attributs ou caractéristiques .

Les problèmes d'apprentissage se répartissent en plusieurs catégories :

- Apprentissage supervisé : dans lequel les données sont accompagnées d'attributs supplémentaires que nous voulons prédire. Ce problème peut être soit :
  - Classification : les échantillons appartiennent à deux ou plusieurs classes et nous voulons apprendre à partir de données déjà étiquetées comment prédire la classe des données non étiquetées. Un exemple de problème de classification serait la reconnaissance de chiffres manuscrits, dans laquelle le but est d'affecter chaque vecteur d'entrée à l'une d'un nombre fini de catégories discrètes. Une autre façon de concevoir la classification est une forme discrète (par opposition à continue) d'apprentissage supervisé où l'on a un nombre limité de catégories et pour chacun des n échantillons fournis, on est d'essayer de les étiqueter avec la bonne catégorie ou classe .
  - Régression : si la sortie souhaitée est constituée d'une ou plusieurs variables continues, alors la tâche est appelée régression . Un exemple de problème de régression serait la prédiction de la longueur d'un saumon en fonction de son âge et de son poids.
- Apprentissage non-supervisé : dans lequel les données d'apprentissage consistent en un ensemble de vecteurs d'entrée X sans aucune valeur cible correspondante.
  - Clustering : Le but de tels problèmes peut être de découvrir des groupes d'exemples similaires dans les données
  - Estimation de densité : déterminer la distribution des données dans l'espace d'entrée
  - projeter les données à partir d'un espace jusqu'à deux ou trois dimensions à des fins de visualisation.

Les Données

Données d'entrainement et données de Test

L'apprentissage automatique consiste à apprendre certaines propriétés d'un ensemble de données, puis à tester ces propriétés par rapport à un autre ensemble de données. Une pratique courante en apprentissage automatique consiste à évaluer un algorithme en divisant un ensemble de données en deux. Nous appelons l'un de ces ensembles l' ensemble d'apprentissage, sur lequel nous apprenons certaines propriétés ; nous appelons l'autre ensemble l'ensemble de test, sur lequel nous testons les propriétés apprises.

Les données traités par la librairie sklearn sont des donnés tabulaires. Ces données sont représentées par des tableaux à 2 dimensions :

- Les lignes représentent les enregistrements,
- Les colonnes les attributs qui caractérisent chacun des enregistrements

Une donnée (un enregistrement) est un vecteur de caractéristiques (valeurs des attributs), généralement des réels, mais des entiers, booléens. Dans le cas de l'apprentissage supervisé, un attribut spécifique correspond au label d'un enregistrement (son gold-standard). Les labels peuvent être de différents types, généralement ce sont des valeurs discrètes (des entiers ou chaînes de caractères). Chaque valeur correspond alors à l'identifiant d'une classe. Dans le cas de problème de régressions, il est possible d'avoir des valeurs continue. Les labels sont contenus dans un tableau à une dimension, sauf rares cas où ils peuvent être dans le vecteur de paramètres (classification multi-objective).

Pour sklearn les données peuvent être sous deux formes usuelles : des tableaux Numpy ou Pandas. D'un point de vue de la programmation, le travail principal consiste souvent à transformer les donnés brutes en entrée sous la forme d'un tableau qui sera exploitable par les algorithmes de sklearn.

Plus concrètement :
Scikit-learn traite des informations d'apprentissage à partir d'un ou plusieurs ensembles de données qui sont représentés sous forme de tableaux 2-D. Ils peuvent être compris comme une liste d'observations multidimensionnelles. On dit que le premier axe de ces tableaux est l' axe des *chantillons*, tandis que le second est l' axe des *caractristiques*.

```
# Un exemple simple  avec scikit-learn : jeu de donnees iris
from sklearn import datasets
iris = datasets.load_iris()
data = iris.data
data.shape
```

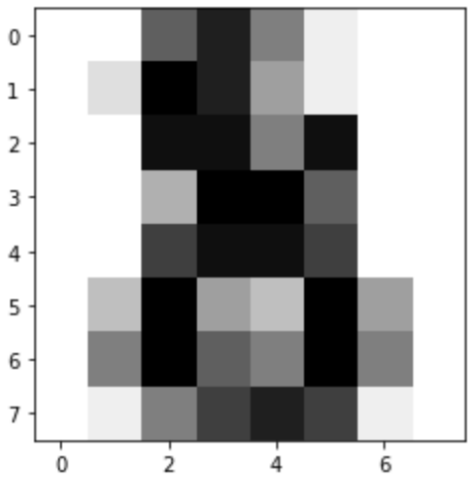
On obtient (150, 4) : ce qui correspond à 150 observations d'iris, chacune décrite par 4 caractéristiques : la longueur et la largeur de leurs sépales et pétales, comme détaillé dans iris.DESCR.

Lorsque les données ne sont pas initialement dans la forme, elles doivent être prétraitées pour être utilisées par scikit-learn. $(n_samples, n_features)$

Voyant maintenant un exemple de remodelage des données. Livré égalemtn avec sklearn, l'ensemble de données de chiffres (Digits) est composé de 1797 images $8x8$ de chiffres écrits à la main

```
digits = datasets.load_digits()
digits.images.shape

import matplotlib.pyplot as plt
plt.imshow(digits.images[-1],
           cmap=plt.cm.gray_r)
```

```
Out[42]: <matplotlib.image.AxesImage at 0x7ff112775190>
```



Pour utiliser cet ensemble de données avec scikit-learn, nous transformons chaque image $8x8$ en un vecteur de caractéristiques de longueur 64.

```
data = digits.images.reshape(
    (digits.images.shape[0], -1))
```

Voilà, maintenant nos données sont utilisable par sklearn, il suffit juste de faire un simple $print(data)$ pour voir le contenu du tableau correspondant à la dernière image affichée ci-dessus.

Pour charger à partir d'un jeu de données externe, scikit-learn fonctionne sur toutes les données numériques stockées sous forme de tableaux numpy ou de matrices scipy sparse. D'autres types convertibles en tableaux numériques tels que les pandas DataFrame sont également acceptables.

Voici quelques méthodes recommandées pour charger des données en colonnes standard dans un format utilisable par scikit-learn :

- scikit-learn *datasets.load_files* : pour les répertoires de fichiers texte
- scikit-learn *datasets.load_svmlight_file* : pour le format svmlight ou libSVM sparse
- *pandas.io* : pour lire des données sous format : CSV, Excel, etc.
- *numpy/routines.io* : pour le chargement standard de données en colonnes dans des tableaux numpy

En pratique, pour des images satellites, les tableaux de données pourrait assez naturellement être construit sous une forme tabulaire en prenant chaque pixel comme enregistrement avec les valeurs des bandes comme attributs. Le travail principal à réaliser est donc de transformer une source de données sous une forme tableau de données pandas (par exemple).

La librairie sklearn

Scikits-learn est une librairie d'apprentissage automatique couvrant une très grande gamme des tâches usuelles de cette discipline : apprentissage supervisé, non supervisé, par renforcement,... On y retrouve les algorithmes classiques tels que :

- non-supervisé
  - K-means (partitionnement en k-moyennes)
  - DBScan (algorithme de partitionnement de données)
- supervisé
  - Linear Regression (régression linéaire)
  - Decision Tree (arbres de décision)
  - SVM (machines à vecteur de support)
  - KNN (plus proches voisins)
  - Naive Bayes (classification naive bayésienne)

Premiers pas : sklearn

La librairie sklearn est très bien pensée pour faciliter le travail de la mise en place d'une chaîne de traitement incluant le chargement de donnés, sa sélection et l'affichage de résultat. Pour cela il se connecte facilement à d'autres librairies python également classiques dans le domaine des data science avec Python (numpy, pandas, matplotlib).

scikit-learn est livré avec quelques ensembles de données standard, par exemple les ensembles de données sur l' iris et les chiffres pour la classification et l' ensemble de données sur le diabète pour la régression.

Les trois instructions suivantes nous permettent de charger deux ensembles de données *Iris* et *Digits*.

```
from sklearn import datasets
# chargement des ensembles de donnees Iris et Digits
iris = datasets.load_iris()
digits = datasets.load_digits()
```

Un ensemble de données est un objet de type dictionnaire qui contient toutes les données et certaines métadonnées sur les données. Ces données sont stockées dans le *.data* membre, qui est un tableau. Dans le cas d'un problème supervisé, une ou plusieurs variables de réponse sont stockées dans le membre.

Par exemple, dans le cas du jeu de données digits, *digits.data* donne accès aux fonctionnalités qui peuvent être utilisées pour classer les échantillons digits :

```
print(digits.data)
[[ 0.   0.   5. ...   0.   0.   0.]
 [ 0.   0.   0. ...  10.   0.   0.]
 [ 0.   0.   0. ...  16.   9.   0.]
 ...
 [ 0.   0.   1. ...   6.   0.   0.]
 [ 0.   0.   2. ...  12.   0.   0.]
 [ 0.   0.  10. ...  12.   1.   0.]]
```

Quant au fichier *digits.target*, il donne la vérité pour l'ensemble de données numériques, c'est-à-dire le nombre correspondant à chaque image numérique que nous essayons d'apprendre :

```
digits.target
 on obtient :
array([0, 1, 2, ..., 8, 9, 8])
```

Le tableaux de données est de la forme : liste 2D. Les données sont toujours un tableau 2D, shape , bien que les données d'origine puissent avoir une forme différente. Dans le cas des chiffres, chaque échantillon original est une image de forme et peut être consulté en utilisant : (n-samples, n-features) (8, 8) : un tableau de 2 dimensions.

```
digits.images[0]
 on obtient :
array([[  0.,   0.,   5.,  13.,   9.,   1.,   0.,   0.],
       [  0.,   0.,  13.,  15.,  10.,  15.,   5.,   0.],
       [  0.,   3.,  15.,   2.,   0.,  11.,   8.,   0.],
       [  0.,   4.,  12.,   0.,   0.,   8.,   8.,   0.],
       [  0.,   5.,   8.,   0.,   0.,   9.,   8.,   0.],
       [  0.,   4.,  11.,   0.,   1.,  12.,   7.,   0.],
       [  0.,   2.,  14.,   5.,  10.,  12.,   0.,   0.],
       [  0.,   0.,   6.,  13.,  10.,   0.,   0.,   0.]])
```

Avec l'exemple suivant on va illustrer comment à partir du problème d'origine, on peut façonner les données pour la consommation dans scikit-learn.

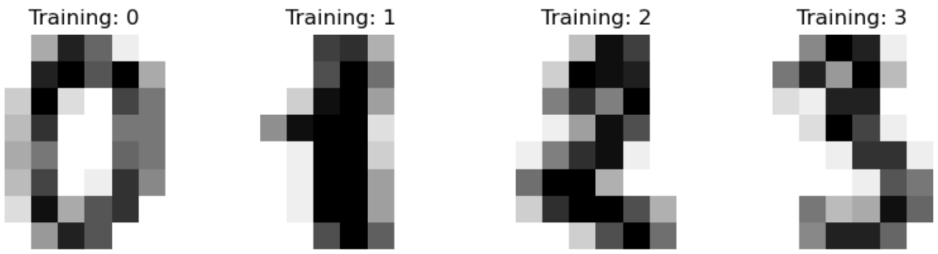
Reconnaissance des chiffres écrits à la main

La librairie scikit-learn peut être utilisé pour reconnaître des images de chiffres manuscrits, de 0 à 9.

```
# Importation de matplotlib
import matplotlib.pyplot as plt

# Importation de : datasets, le  classifieurs et performance
# metrics
from sklearn import datasets, svm, metrics

# importation de la fonction qui fait le split des donnes
from sklearn.model_selection import train_test_split
```

Notre ensemble de données de chiffre se compose d'images de chiffres 8*x*8 pixels.
L'attribut *images* de l'ensemble de données stocke des tableaux 8*x*8 de valeurs en niveaux de gris pour chaque image. Nous utiliserons ces tableaux pour visualiser les 4 premières images. L'attribut *target* de l'ensemble de données stocke le chiffre que chaque image représente et cela est inclus dans le titre des 4 graphiques ci-dessous.

```
digits = datasets.load_digits()

_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, label in zip(axes, digits.images, digits.target):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Training: %i' % label)
```



Pour appliquer un classificateur sur ces données, nous devons aplatir les images, en transformant chaque tableau 2-D de valeurs en niveaux de gris de shape $(8,8)$ en shape $(64,)$.

Par la suite, l'ensemble de données complet aura la forme : $(n_samples, n_features)$, où $n_samples$ est le nombre d'images et $n_features$ est le nombre total de pixels dans chaque image

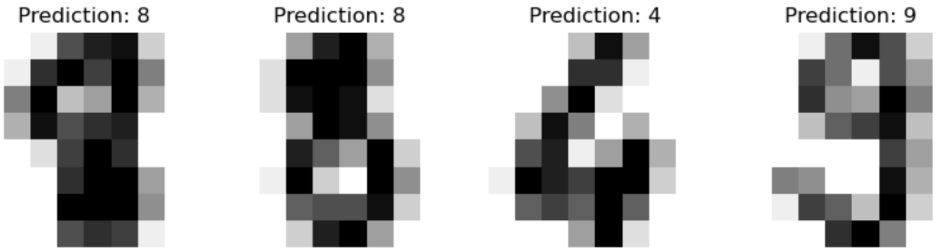
```
# aplatir les  images
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))
```

Ensuite nous divisons les données en sous-ensembles de train (apprentissage) et de test et ajuster un classificateur de vecteur de support (SVM) sur les échantillons de train. Le classificateur ajusté peut ensuite être utilisé pour prédire la valeur du chiffre pour les échantillons du sous-ensemble de test.

```
# Creation du classifieur: svm
clf = svm.SVC(gamma=0.001)

# Split nos donnees en 50% train anetd 50% test
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False)

# apprentissage des chiffres sur le sous-ensemble train
clf.fit(X_train, y_train)

# Predire la caleur du chiffre du sous-ensemble test
predicted = clf.predict(X_test)
```

Maintenant que notre classificateur (SVM) a appris sur le sous-ensemble de donnée, et fait une prédiction sur l'autre sous-ensemble. Nous visualisons les 4 premiers échantillons de test et montrons leur valeur numérique prédite dans le titre.

```
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, prediction in zip(axes, X_test, predicted):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title(f'Prediction: {prediction}')
```



La première et la deuxième image sont reconnues comme étant le chiffre 8, la troisième est le chiffre 4, la dernière comme 9.

La fonction *classification_report* de *metrics* crée un rapport texte montrant les principales mesures de classification effectuées sur nos données.

```
print(f"Classification report for classifier {clf}:\n"
      f"{metrics.classification_report(y_test, predicted)}\n")
```

on obtient :

```
Classification report for classifier SVC(gamma=0.001):
              precision    recall  f1-score   support

           0       1.00      0.99      0.99        88
           1       0.99      0.97      0.98        91
           2       0.99      0.99      0.99        86
           3       0.98      0.87      0.92        91
           4       0.99      0.96      0.97        92
           5       0.95      0.97      0.96        91
           6       0.99      0.99      0.99        91
           7       0.96      0.99      0.97        89
           8       0.94      1.00      0.97        88
           9       0.93      0.98      0.95        92

    accuracy                           0.97       899
   macro avg       0.97      0.97      0.97       899
weighted avg       0.97      0.97      0.97       899
```

Nous pouvons également tracer une matrice de confusion des valeurs numériques réelles

(les lignes) et des valeurs numériques prédites (les colonnes).

```
disp = metrics.confusion_matrix(y_test, predicted)
print(f"Confusion matrix:\n{disp}")

on obtient :

Confusion matrix:
[      0   1   2   3   4   5   6   7   8   9
      ---------------------------------------
 0 : [87   0   0   0   1   0   0   0   0   0]
 1 : [ 0  88   1   0   0   0   0   0   1   1]
 2 : [ 0   0  85   1   0   0   0   0   0   0]
 3 : [ 0   0   0  79   0   3   0   4   5   0]
 4 : [ 0   0   0   0  88   0   0   0   0   4]
 5 : [ 0   0   0   0   0  88   1   0   0   2]
 6 : [ 0   1   0   0   0   0  90   0   0   0]
 7 : [ 0   0   0   0   0   1   0  88   0   0]
 8 : [ 0   0   0   0   0   0   0   0  88   0]
 9 : [ 0   0   0   1   0   1   0   0   0  90]]
```

Exercice

En s'inspirant de l'exemple précédant, répondez aux questions suivantes :

- question 1 :
  - Mettre que 40% des images pour l'ensemble d'apprentissage
  - Afficher les 6 premières images de l'ensemble test avec leurs prédictions
  - Afficher sa matrice de confusion.
- question 2 :
  - Mettre que 80% des images pour l'ensemble de test
  - Afficher les 6 premières images de l'ensemble test avec leurs prédictions
  - Afficher sa matrice de confusion.
  -
- question 3 :
  - Utiliser toutes les images pour l'ensemble train sauf la dernière image
  - Afficher la dernière (et la seule) image de l'ensemble test avec sa prédiction