M2 — SIIA

TP Interactive Machine Learning — 2h

Cédric Buche

Contents

1 Objectifs

Le *Machine Learning* est une façon de modéliser un phénomène dans le but de prendre une décision ou de prédire une valeur. Le principe est de construire une représentation des données de façon automatique en entraînant notre machine sur un jeu de données de référence, c'est la phase d'apprentissage. Cette machine est ensuite utilisée pour prédire des valeurs ou identifier des propriétés. Les points importants de cette méthode sont :

- ▷ En premier lieu, les données, c'est la base ! Plus elles seront propres et comprises, plus la suite sera facile. Dans le cadre de cet article, les données ont déjà été traitées et nettoyées, les traitements supplémentaires que nous allons faire auront seulement pour but d'adapter au mieux nos données aux algorithmes ;
- ▷ La tâche à accomplir : comme pour les données, plus elle sera claire et précise, plus la suite sera facile. Dans le cadre de cet article, la tâche est d'effectuer une classification supervisée des objets identifiant le type de galaxie ;
- ▷ L'algorithme d'apprentissage, à choisir en fonction des données, de leur dimension, de leur quantité, mais aussi en fonction de la tâche à accomplir, régression ou classification, supervisé ou non-supervisé. Dans le cadre de cet article, plusieurs algorithmes seront testés pour avoir une vision globale des possibilités du machine learning ;
- ▷ L'évaluation des performances du modèle : cette évaluation est nécessaire pour valider le modèle et pour comparer plusieurs algorithmes. Dans le cadre de cet article, plusieurs tests seront définis pour comparer les algorithmes utilisés.

2 Classification des galaxies avec du ML

Depuis l'antiquité, les astronomes ont fourni une classification des objets du ciel, en partant des plus brillants, observés à l'œil nu, il y a plus de 5000 ans, vers les plus difficiles à observer, même avec un télescope de 8m de nos jours. Si le ciel est dominé par les étoiles de notre propre galaxie, la "Voie lactée", et les planètes de notre système solaire, il a fallu attendre les premiers télescopes pour accéder au monde un peu plus obscur, de faible luminosité, et observer les galaxies. Messier, en 1774, édite un catalogue de 110 objets diffus, qui en fait le premier référencement de nébuleuses et de galaxies. En 1936, E. Hubble (celui qui donna son nom au télescope spatial américain) a proposé une classification des galaxies en proposant un diagramme d'évolution [2] (figure 1). Ce diagramme a évolué durant les années (de Vaucouleurs, 1959 et 1964), Kormendy et Bender (1996), et d'autres par la suite, mais il reste encore la base des systèmes de classification d'aujourd'hui.

La classification des galaxies est une étape complexe qui vise à rassembler les objets par des caractéristiques communes dans des catégories fondamentales. Elle fait appel à des processus de décision qui implique une analyse de la morphologie, de la nature des objets, mais parfois aussi une analyse multidimensionnelle (temporelle et en longueur d'onde). Parfois même la classification dépend de la personne classant les données.

En restant synthétique, l'ensemble des galaxies peut être classé selon trois grandes catégories (voir figure 1), elliptiques (E), Spirales (S) (normales ou barrées) et Irrégulières (Irr). C'est cette approche (simpliste) que nous allons utiliser pour cette initiation. Nous allons dans

la suite essayer de traiter automatiquement les nouvelles images afin de définir de classer les galaxies dans ces 3 classes.

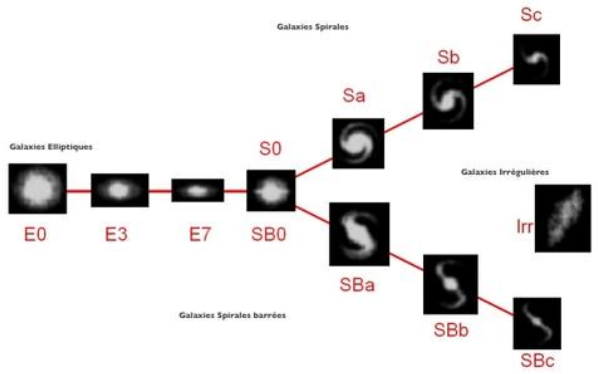

Figure 1: Séquence de Hubble (http://astronomy.swin.edu.au/cosmos/H/Hubble+Classification)

3 Install

Pour les ordinateurs perso :

```
$ pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose
$ pip install --user scikit-learn
$ pip install --user keras tensorflow np_utils
$ pip install --user fitsio
```

4 Data

4.1 Raw Data

Nous allons travailler avec les données astrophysiques du projet COSMOS (Cosmic Evolution Survey). Ce sondage a été défini par une collaboration de 200 scientifiques conduits par Peter Capak pour résoudre les problématiques de formation et d'évolution des galaxies. Il couvre une région d'environ 2 degrés carrés qui a été couverte par un ensemble de télescopes donnant une couverture unique en longueur d'onde et en précision d'un échantillon de près de deux millions de galaxies.

Nous allons nous limiter à une partie de quelques milliers de galaxies qui ont été classées visuellement. Ces données sont disponibles sur ..... Dans le dossier vous devriez

avoir un répertoire `data` contenant :

- ▷ un fichier texte `data-COSMOS-10000-id.txt` de 13000 objets ayant été classés ;
- ▷ un répertoire `image` contenant 3998 images de galaxies déjà classifiées ;
- ▷ un répertoire `new_image` contenant 1000 images de galaxies non classifiées.

Le fichier `iml_student.py` charge les données, l'utilitaire de plot de matplotlib pour vérifier et visualiser les données (figure 2).

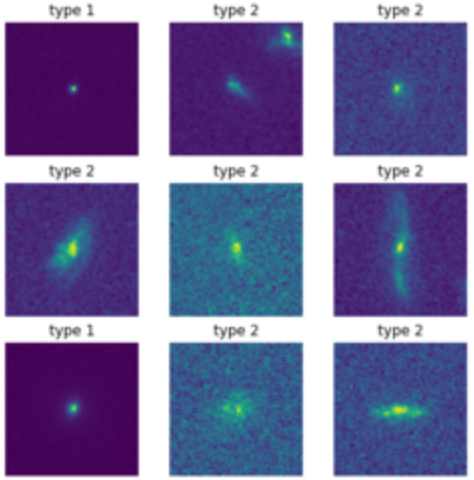

Figure 2: Visualisation des 9 premières images de galaxies de l'échantillon ainsi que du type morphologique associé (1 = "Elliptiques", 2= "Spirales", 3 = "Irrégulières")

4.2 Data Transformation

Bien que les données aient déjà été traitées et nettoyées, deux transformations sont toutefois requises.

D'une part, l'implémentation des algorithmes de scikit-learn s'attend à recevoir en entrée des données standardisées, c'est-à-dire transformées de façon à avoir une moyenne nulle et une variance égale à un. Sans rentrer dans le détail, le but de cette transformation est de permettre une variation de poids uniforme quand on les applique aux données.

D'autre part, la plupart des algorithmes de scikit-learn n'acceptent que des données sous forme de vecteurs en entrée alors que nos images sont des matrices. On peut simplement convertir nos matrices M x N (nos images donc) en vecteur de taille MN. Par exemple, 100

matrices de 100 x 100 pixels chacune, peuvent être converties en 100 vecteurs contenant chacun 10000 éléments.

Le fichier `iml_student.py` effectue ces transformations.

5 Evaluation

Avant de commencer à mettre en place notre "classifier", et pour mieux comprendre les résultats, il est indispensable de parler en quelques mots de l'évaluation des performances.

L'évaluation des performances sert à comparer deux modèles afin de déterminer quel est le meilleur. Cela peut être pour choisir entre deux algorithmes d'apprentissage ou d'optimiser, pour un même algorithme, les différentes valeurs d'hyper-paramètres ou enfin de sélectionner différentes façons de pré-processer les données...

5.1 Cross validation

Une méthode facile à mettre en place, en général, est la validation croisée qui consiste, dans sa version la plus simple, à séparer son échantillon en un jeu d'entraînement et un jeu de test. Le jeu d'entraînement sera utilisé pour entraîner le modèle et le jeu de test pour évaluer les performances. Ce test peut être, toutefois, dépendant de la méthode de séparation de l'échantillon. Il faut s'assurer que la distribution des propriétés des ensembles est bien la même dans l'échantillon initial et dans les jeux d'entraînement et de test. Mais, même dans ce cas, le résultat peut dépendre de la proportion des deux jeux.

Des versions d'évaluation des performances plus évoluées sont souvent utilisées pour pallier à ces différents biais. Par exemple, la validation croisée "k-fold" consiste à séparer l'échantillon initial en k jeux de données, choisir un jeu de test et entraîner sur les (k - 1) autres jeux et répéter l'opération k fois en changeant le jeu de test. Bien que plus fiable, on voit bien que cette méthode nécessite beaucoup plus de temps de calcul. Dans la suite, nous allons nous contenter de la validation croisée avec un jeu d'entraînement et un jeu de test.

5.2 Precision/recall/specificity/f-score

La façon de la plus simple de calculer la performance de l'algorithme dans le cas d'une classification est d'utiliser l'échantillon de test pour prédire la classe avec le modèle entraîné et de comparer les prédictions avec les classes de références.

En ne s'intéressant qu'à un cas binaire, quatre cas sont possibles pour une valeur testée

:

- ▷ le cas vrai positif (VP), la valeur testée est identifiée comme appartenant à la classe et cette valeur appartient bien à la classe ;
- ▷ le cas vrai négatif (VN), la valeur test est identifiée comme n'appartenant pas à la classe et n'appartient effectivement pas à la classe ;
- ▷ le cas faux positif (FP), la valeur test est identifiée comme appartenant à la classe, mais elle n'appartient pas à la classe ;
- ▷ le cas faux négatif (FN), la valeur test est identifiée comme n'appartenant pas à la classe alors qu'elle appartient à la classe.

Toujours dans le cas d'une seule classe, il est possible d'estimer le nombre de vrai/faux positifs/négatifs pour toutes les valeurs de l'échantillon de test et d'obtenir ainsi la matrice de confusion. En appliquant le même principe au cas avec N classes, il y aura $(N-1)$ cas faux différents (en fonction de la classe choisie par erreur) et la matrice de confusion sera de taille NxN.

A partir de ces chiffres, nous pouvons dériver plusieurs estimateurs de performance :

- ▷ la précision = VP / (VP + FP), c'est-à-dire le taux de bonnes réponses sur l'ensemble des réponses mesurées comme vraies
- ▷ la sensibilité (recall en anglais) = VP / (VP + FN), c'est-à-dire le taux de bonnes réponses positives par rapport à l'ensemble des réponses qui sont effectivement vraies
- ▷ la spécificité = VN / (VN + FP), c'est-à-dire le taux de bonne réponse négative par rapport à l'ensemble des réponses qui sont effectivement vraies
- ▷ le f-score = 2 x (précision x sensibilité) / (précision + sensibilité).

Les estimateurs de performance étant multiples, choisir le meilleur modèle n'est pas évident et dépend de l'utilisation qui sera faite du modèle. L'exemple classique, afin de bien comprendre, est celui de la classification des champignons. On veut bien sûr identifier avec la meilleure précision possible les champignons comestibles, mais on préfèrerait éviter de classifier comestible par erreur un champignon hallucinogène et on ne veut surtout pas se tromper sur un champignon mortel !

5.3 ROC

Il existe bien sûr d'autres façons d'estimer les performances que nous n'utiliserons pas. Pour aller plus loin, vous pouvez utiliser la courbe de ROC, comme dans l'exemple de scikit-learn (`http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html`), et les comparaisons aux cas naïfs, aussi illustrés dans la documentation de scikit-learn (`http://scikit-learn.org/stable/auto_examples/calibration/plot_compare_calibration.html`).

6 Classification

6.1 SVM

Nous allons commencer notre étude par l'utilisation d'une machine à vecteur de support (SVM) parce qu'a priori c'est un algorithme simple de prédiction linéaire, c'est-à-dire qu'il calcule une combinaison linéaire (somme pondérée) des variables d'entrée pour prédire la

valeur de sortie.

Dans le cas de la classification, vous pouvez voir ça comme un algorithme qui essaie de séparer vos points par des droites. Ce n'est pas complètement aussi simple. Il existe des hyper-paramètres, c'est-à-dire des paramètres fixés a priori dans le modèle. Par exemple la "marge" permet de donner, en quelque sorte, une largeur aux droites de séparations. Il est aussi possible, par changement de référentiel, de faire des séparations avec des fonctions plus complexes qu'une droite dans le référentiel d'origine. Pour cela, utiliser `scikit-learn`.

Nous pouvons lancer l'apprentissage de notre SVM (cette étape prend quelques minutes). Afficher la matrice de confusion. Afficher la précision, la sensibilité et le f-score.

Les différentes valeurs de précision et de sensibilité, permettent de se faire une idée des résultats croisés par rapport aux différentes valeurs attendues. Les valeurs 1, 2, 3 représentent les classes (Elliptiques, Spirales, Irrégulières). Idéalement, les niveaux de précisions, sensibilités et f-scores de chaque classe devraient être égales à 1. Le support est simplement le nombre d'objets considérés dans les différents calculs.

Il existe une classe dans `scikit-learn` pour optimiser les hyper-paramètres : `GridSearchCV`. Attention, toutefois, cela prend typiquement 20 minutes avec 8 jobs tournant en parallèle (1 par processeur) sur un laptop. À ne lancer que si vous avez du temps ou une machine assez puissante.

6.2 Random Forest

Le principe des méthodes ensemblistes est de combiner statistiquement plusieurs modèles pour prédire un résultat plus robuste. Ces méthodes se décomposent en deux catégories : les méthodes dites parallèles, qui entraînent les modèles indépendamment et les méthodes séquentielles, aussi appelées "boost", qui génèrent les modèles de façon itérative en pondérant les données qui sont sources d'erreurs.

Tester le *Random Forest*

6.3 Boost-tree

Tester *AdaBoostClassifier* puis *GradientBoostingClassifier*

6.4 Neural Networks

Un neurone artificiel est une fonction mathématique qui reçoit une ou plusieurs entrées et les additionne pour produire une sortie. Habituellement, chaque entrée est pondérée séparément et la somme est filtrée par une fonction non linéaire appelée fonction d'activation. Un réseau de neurones (artificiels) est donc un ensemble de neurones connectés. Chaque connexion entre les neurones peut transmettre un signal à un autre neurone. Le neurone de réception peut traiter les signaux qui lui sont envoyés et transmet sa sortie aux neurones qui lui sont connectés en aval dans le réseau.

Le réseau de neurones le plus simple est le perceptron composé d'une couche d'entrée avec un neurone par entrée tous reliés à un neurone de sortie. Ce réseau permet de représenter des modèles basés sur une combinaison linéaire des variables. Pour des modèles plus complexes, on peut ajouter des couches intermédiaires, appelées couches cachées. Chaque neurone d'une couche est connecté à tous les neurones de la couche au-dessus de lui. C'est ce qu'on appelle le perceptron multi-couches (ou multilayer perceptron en anglais, "MLP"). Empiler des perceptrons en un réseau de neurones multi-couches permet de modéliser des fonctions arbitrairement complexes. C'est ce qui donne aux réseaux de neurones profonds la puissance prédictive qui fait actuellement leur succès. Mais plus il y a de paramètres, plus la quantité de données d'apprentissage nécessaire afin de définir les valeurs de ces paramètres sans risquer le sur-apprentissage, est important. Il existe de nombreuses autres architectures de réseaux de neurones.

Entraînons donc un MLP sur nos images de galaxies. Commençons sagement avec une couche cachée de 100 neurones.

7 Prediction

À l'aide des modèles précédents, vous êtes donc en mesure de prédire le type d'une galaxie à partir de son image. Et ça tombe bien, on vient de recevoir un nouvel échantillon ! Vous pouvez le retrouver dans votre sous-répertoire `<myDirectory>/data/new_image/`.

Utiliser la méthode `predict()` sur les modèles précédents. Ajouter une visualisation. Conclusion?