Lab - Machine Learning - Classification — 2x2h

Cédric Buche

Contents

1 Context

1.1 Classification of Galaxies

Astronomers have provided a classification of objects from the sky. Messier, in 1774, publishes a catalog of 110 diffuse objects, which makes it the first referencing of nebulae and galaxies. In 1936, E. Hubble (who gave his name to the American space telescope) proposed a classification of galaxies by proposing a diagram of evolution (figure 1). This diagram evolved during the years (de Vaucouleurs, 1959 and 1964), Kormendy and Bender (1996), and others thereafter, but it still remains the basis of today's classification systems.
The classification of galaxies is a complex step that aims to bring together objects by common features in fundamental categories. It uses decision-making processes that involve an analysis of the morphology, the nature of the objects, but sometimes also a multidimensional analysis (temporal and wavelength). Sometimes even the classification depends on the person filing the data.
By remaining synthetic, all galaxies can be classified according to **three broad categories (see figure 1), elliptic (E), Spirals (S) (normal or crossed out) and Irregular (Irr)**. It is this (simplistic) approach that we will use for this initiation. We will then try to automatically process the new images to define how to classify the galaxies in these 3 classes.

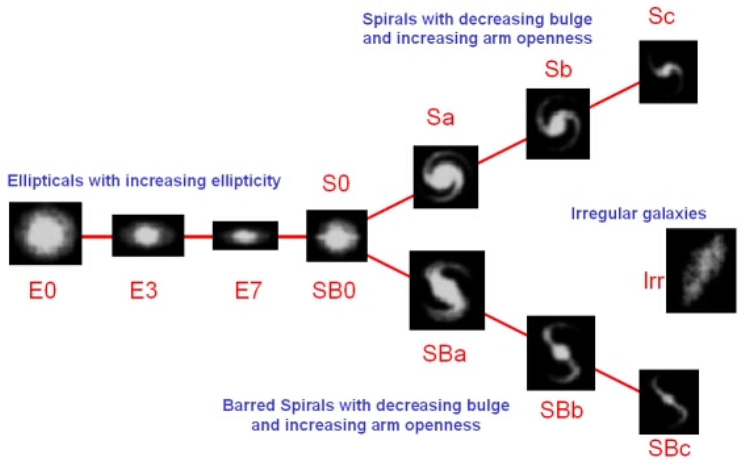
Spirals with decreasing bulge and increasing arm openness
Sc
Sb
Sa
Ellipticals with increasing ellipticity
S0
Irregular galaxies
Irr
E0
E3
E7
SB0
SBa
SBb
SBc
Barred Spirals with decreasing bulge and increasing arm openness

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

1.2 Install

```
$ 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
```

1.3 Data

1.3.1 Raw Data

We will work with the astrophysical data of the COSMOS project (Cosmic Evolution Survey). This survey was defined by a collaboration of 200 scientists led by Peter Capak to solve the problems of formation and evolution of galaxies.

We will limit ourselves to a part of a few thousand galaxies that have been classified visually. These data are available on `/home/TP/modules/sujets/IML/tp_galaxies/data`.

In the folder you should have a `data` directory containing:

- ▷ a text file `data-COSMOS-10000-id.txt` of 13000 classified objects;
- ▷ a `image` directory containing 3998 images of already classified galaxies;
- ▷ a `new_image` directory containing 1000 images of unclassified galaxies.

The `/home/TP/modules/sujets/IML/tp_galaxies/iml_student.py` file loads the data. It also makes it possible to visualize them (figure 2).

1.3.2 Data Transformation

Although the data has already been processed and cleaned up, two transformations are required.

On the one hand, the implementation of the `scikit-learn` algorithms is expected to receive standardized data input, that is, transformed to have a zero mean and a variance equal to one. Without going into detail, the goal of this transformation is to allow uniform weight variation when applied to data.

On the other hand, most `scikit-learn` algorithms only accept input vector data while our images are matrices. We can simply convert our matrices M x N (our images therefore) into vectors of size MN. For example, 100 matrices of 100 x 100 pixels each, can be converted into 100 vectors each containing 10,000 elements.

The `iml_student.py` file performs these transformations.

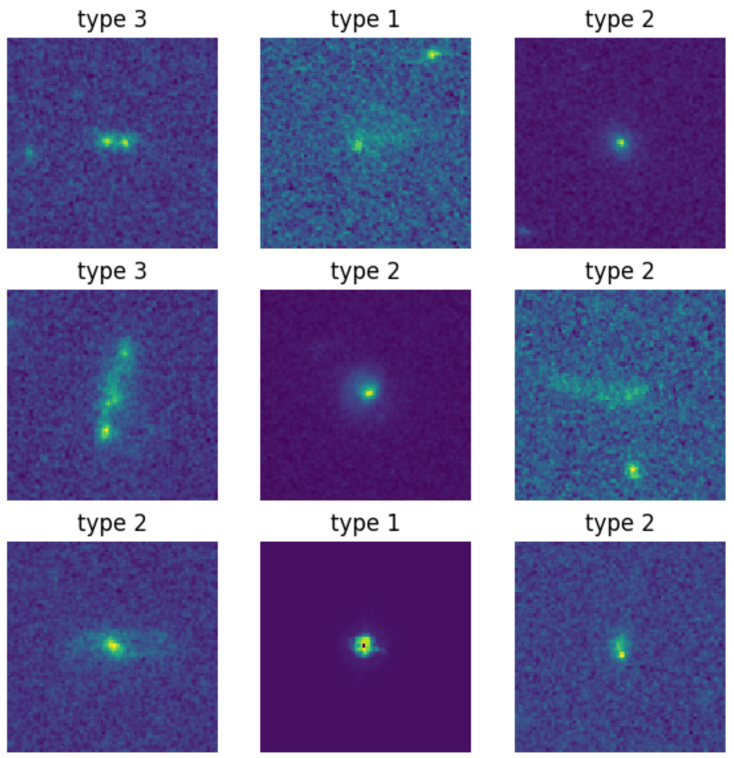

Figure 2: Visualization of the first 9 images of galaxies of the sample as well as the associated morphological type (1 = ”Elliptic”, 2 = ”Spirals”, 3 = ”Irregular”)

1.4 Evaluation

Before we begin to set up our "classifier", and to better understand the results, it is essential to speak in a few words about performance evaluation.

Performance evaluation is used to compare two models to determine which is the best. This can be to choose between two learning algorithms or to optimize, for the same algorithm, the different values of hyper-parameters or finally to select different ways of pre-processing the data ...

1.4.1 Cross validation

An easy method to set up, in general, is cross-validation which consists, in its simplest version, in separating its sample into a training game and a test game. The training game will be used to train the model and the test game to evaluate performance. This test may, however, be dependent on the method of separation of the sample. It must be ensured that the distribution of the properties of the sets is the same in the initial sample and in the training and test games. But even in this case, the result may depend on the proportion of the two games.

More advanced performance evaluation versions are often used to overcome these different biases. For example, cross-validation "k-fold" consists in separating the initial sample into k datasets, choosing a test set and training on the (k - 1) other games and repeating the operation k times by changing the set of test. Although more reliable, it is clear that this method requires much more computing time. In the following, we will be satisfied with the cross validation with a training game and a test game.

1.4.2 Precision/recall/specificity/f-score

The simplest way to calculate the performance of the algorithm in the case of a classification is to use the test sample to predict the class with the trained model and to compare the predictions with the reference classes.

By only looking at a binary case, four cases are possible for a value tested:

- ▷ the true positive case (VP), the tested value is identified as belonging to the class and this value belongs to the class;
- ▷ the true negative case (VN), the test value is identified as not belonging to the class and does not actually belong to the class;
- ▷ the false positive case (FP), the test value is identified as belonging to the class, but it does not belong to the class;
- ▷ the false negative case (FN), the test value is identified as not belonging to the class while it belongs to the class.

In the case of a single class, it is possible to estimate the number of true / false positives

/ negatives for all the values of the test sample and thus obtain the confusion matrix. Applying the same principle to the case with N classes, there will be $(N - 1)$ different false cases (depending on the class chosen by mistake) and the confusion matrix will be of size NxN.

From these numbers, we can derive several performance estimators:

- ▷ accuracy = VP / (VP + FP), that is, the rate of correct answers on all responses measured as true
- ▷ the sensitivity = VP / (VP + FN), that is to say the rate of good positive answers in relation to all the answers that are actually true
- ▷ the specificity = VN / (VN + FP), that is to say the rate of good negative answer with respect to all the answers which are actually true
- ▷ the f-score = 2 x (precision x sensitivity) / (precision + sensitivity).

Since performance estimators are multiple, choosing the best model is not obvious and depends on how the model is used.

1.4.3 ROC

There are of course other ways to estimate the performance that we will not use. To go further, you can use the ROC curve, as in the example of `scikit-learn` (`http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html`), and comparisons to naïve cases, also illustrated in the literature of `scikit-learn` (`http://scikit-learn.org/stable/auto_examples/calibration/plot_compare_calibration.html`).

2 Work

2.1 Learning

2.1.1 SVM

We will start our study with the use of a support vector machine (SVM) because a priori it is a simple algorithm of linear prediction, that is to say it calculates a linear combination (weighted sum) of the input variables to predict the output value. In the case of classification, you can see this as an algorithm that tries to separate your points by lines. It's not completely that simple. There are hyper-parameters, that is to say, parameters fixed a priori in the model. For example, the "margin" makes it possible, in a way, to give a width to the separation lines. It is also possible, by changing the repository, to make separations with more complex functions than a straight line in the original repository. For that, use `scikit-learn`.

Question :
Start learning our SVM (this step takes a few minutes).
Show the confusion matrix.
Display accuracy, sensitivity and f-score.

The different values of precision and sensitivity make it possible to get an idea of the results crossed compared to the different expected values. The values 1, 2, 3 represent classes (Elliptic, Spiral, Irregular). Ideally, the levels of precisions, sensitivities and f-scores of each class should be equal to 1. The support is simply the number of objects considered in the different calculations.

There is a class in `scikit-learn` to optimize hyper-parameters: `GridSearchCV`. Be careful, however, it typically takes 20 minutes with 8 jobs running in parallel (1 per processor) on a laptop. Only launch if you have enough time or a powerful machine.

2.1.2 Random Forest

The principle of set methods is to statistically combine several models to predict a more robust result. These methods break down into two categories: the so-called parallel methods, which drive the models independently and the sequential methods, also called "boost", which generate the models iteratively by weighting the data that are sources of errors.

Question :
Test *Random Forest*

Random Forest is one of the most popular and most powerful machine learning algorithms. It is a type of ensemble machine learning algorithm called Bootstrap Aggregation or bagging. The bootstrap is a powerful statistical method for estimating a quantity from a data sample. Such as a mean. You take lots of samples of your data, calculate the mean, then average all of your mean values to give you a better estimation of the true mean value. In bagging, the same approach is used, but instead for estimating entire statistical models, most commonly decision trees. Multiple samples of your training data are taken then models are constructed for each data sample. When you need to make a prediction for new data, each model makes a prediction and the predictions are averaged to give a better estimate of the true output value. Random forest is a tweak on this approach where decision trees are created so that rather than selecting optimal split points, suboptimal splits are made by introducing randomness. The models created for each sample of the data are therefore more different than they otherwise would be, but still accurate in their unique and different ways. Combining their predictions results in a better estimate of the true underlying output value. If you get good results with an algorithm with high variance (like decision trees), you can often get better results by bagging that algorithm.

2.1.3 Boost-tree

Question :
Test *AdaBoostClassifier* and then *GradientBoostingClassifier*

Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added. AdaBoost was the first really successful boosting algorithm developed for binary classification. It is the best starting point for understanding boosting. Modern boosting methods build on AdaBoost, most notably stochastic gradient boosting machines. AdaBoost is used with short decision trees. After the first tree is created, the performance of the tree on each training instance is used to weight how much attention the next tree that is created should pay attention to each training instance. Training data that is hard to predict is given more weight, whereas easy to predict instances are given less weight. Models are created sequentially one after the other, each updating the weights on the training instances that affect the learning performed by the next tree in the sequence. After all the trees are built, predictions are made for new data, and the performance of each tree is weighted by how accurate it was on the training data. Because so much attention is put on correcting mistakes by the algorithm it is important that you have clean data with outliers removed.

2.1.4 Neural Networks

An artificial neuron is a mathematical function that receives one or more inputs and adds them together to produce an output. Usually, each entry is weighted separately and the sum is filtered by a non-linear function called the activation function. A network of (artificial) neurons is therefore a set of connected neurons. Each connection between the neurons can transmit a signal to another neuron. The receiving neuron can process the signals sent to it and transmit its output to the neurons connected to it downstream in the network. The simplest neural network is the perceptron composed of an input layer with an input neuron all connected to an output neuron. This network makes it possible to represent models based on a linear combination of the variables. For more complex models, we can add intermediate layers, called hidden layers. Each neuron in a layer is connected to all the neurons in the layer above it. This is called the multilayer perceptron (or multilayer perceptron in English, "MLP"). Stacking perceptrons into a multi-layer neural network can model arbitrarily complex functions. This is what gives deep neural networks the predictive power that is currently making them successful. But the more parameters there are, the greater the amount of training data needed to define the values of these parameters without risking over-learning. There are many other architectures of neural networks.

Question :
Let's train an MLP on our images of galaxies. Let's start wisely with a hidden layer of 100 neurons.

2.2 Prediction

Using the previous models, you are able to predict the type of a galaxy from its image. And that's good, we just received a new sample! You can find it in your subdirectory `/home/TP/modules/sujets/IML/tp_galaxies/new_image/`

Question :
Use the `predict()` method on previous models.
Add a visualization.
Conclusions?