Lab : Data Mining —

Cédric Buche

Objectives

1. Use machine learning librairies in Python.
2. Load a dataset and understand its structure.
3. Create machine learning models (classification / regression) and analyze results.

Library

In this lab, we will use Python libraries :

1. `Scikit-learn`: machine learning algorithms
2. `NumPy`: Base n-dimensional array package
3. `Matplotlib`: Comprehensive 2D/3D plotting

Documentation

- ▷ Tutorial `http://scikit-learn.org/stable/tutorial/basic/tutorial.html`
- ▷ User Guide `http://scikit-learn.org/stable/user_guide.html`
- ▷ API Reference `http://scikit-learn.org/stable/modules/classes.html`

Contents

1 Dimension reduction - PCA

Dimension reduction is the process of reducing the number of variables by obtaining a set of principal variables.

The main linear technique for dimensionality reduction is Principal Component Analysis (PCA). It performs a linear mapping of the data to a lower-dimensional space : the variance of the data in the low-dimensional representation is maximized. In practice, the covariance matrix of the data is constructed and the eigenvectors on this matrix are computed. The eigenvectors that correspond to the largest eigenvalues (the principal components) can now be used to reconstruct a large fraction of the variance of the original data. Moreover, the first few eigenvectors can often be interpreted in terms of the large-scale physical behavior of the system. The original space (with dimension of the number of points) has been reduced (with data loss, but hopefully retaining the most important variance) to the space spanned by a few eigenvectors.

Question :

1. Use the `load_breast_cancer()` function to load the `breast_cancer` database
2. Display the number of individuals and attributes contained in this database
3. Use the Numpy library to perform a Principal Component Analysis to divide by 2 the number of attributes. The idea is to diagonalize the covariance matrix via eigenvalues and eigenvectors (you can use the function `np.linalg.eig()`)
4. Use the Scikit library to reiterate this operation on the original database through `PCA()`. Make sure that the matrices obtained are identical (cf. `numpy.allclose()`).

2 Supervised learning

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. The inferred function can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations.

2.1 Classification - SVM

Let's perform a classification with a SVM considering the previous data (initial database, without dimensional reduction). You will use the Scikit library.

Question :

1. NORMALIZATION :
   Make sure attributes (here quantitative) are normalized. If it's not the case, it will be necessary to center and reduce the database using Numpy functions.
2. DIVIDE DATASET :
   Divide the database into 3 parts to get a learning set, a validation set and a test set. The distribution will be carried out in a random way must respect the following ratio: 40% / 30% / 30%. Set the seed of the random function to 'None'.
3. LEARNING :
   At first, we will ignore the validation set. Realize learning and prediction using SVC.
4. TEST :
   Represent the matrix of confusion and display classifier performance (see function `accuracy_score`).

Question :

1. DIMENSION REDUCTION :
   Repeat thoses operations on the database with dimension reduction. Distribute the data in the same way so that test set have the same complexity (the idea is to randomly mix a list index). What can you conclude about the interest of the PCA? In the following of this exercise, we will only use the database on which a dimension reduction has been achieved.
2. HYPER-PARAMETERS :
   As you can see in the Scikit documentation, a SVC has some hyper-parameters. These have an influence on the performance of the model. By means of a grid search that you will code by yourself, you will seek to optimize the following two hyper-parameters: cost and gamma. You will use a Gaussian kernel (RBF) function. Rather than keeping only the best classifier, you keep the performances obtained by all the configurations tested. You will need, for finish, represent these performances using a heatmap.

For your information: there are a lot of optimization methods. In the SVMs, which have only a few hyper-parameters, it is not necessary to turn to meta-heuristics such as genetic algorithms or simulated annealing. Very simplistic techniques such as a search by grid or again a random search turn out to be as effective.

IMBALANCED DATABASE :

When it is desired to respond to a classification task, it may happen that some classes are under-represented compared to others. This imbalance consequence of negatively impacting the performance of the model. In order to overcome this problem, there are several techniques:

- ▷ It is possible to oversample or downsample the database in order to to remove this imbalance.
- ▷ It is possible to weight individuals to restore balance during the phase learning.

Question :
In this exercise, we will focus on the second point. As a first step, you will build a new database using the next code:

```
database = datasets.make_classification (n_samples = 100, n_features =
250, n_classes = 2, weights = [0.7,0.3])
```

You will train an SVC with and without the `sample_weight` parameter to understand the importance of the latter in this type of problem. Be carefull, the imbalance of classes must be taken into account in the performance calculation.

2.2 Regression - KNN

Regression is the problem of predicting a continuous quantity output for an example. Regression predictive modeling is the task of approximating a mapping function (f) from input variables (X) to a continuous output variable (y). A continuous output variable is a real-value, such as an integer or floating point value. These are often quantities, such as amounts and sizes.

Question :

1. LAOD/NORMALIZED/DIVIDE DATA SETS :
   Use the `load_boston()` function to load the `boston()` database that is internal to Scikit. In a similar way to previous exercises, normalize the database and divide it into three data sets (ratio 40%, 30%, 30%).
2. TEST :
   As we have discussed in the course, it is possible to evaluate the performance of a regression model through a linear regression (between the random variable and its estimator) and degree of correlation. You will put in place the code to estimate the performance of a regression model. For this do, you will use the Matplotlib library to plot the linear regression as well what are the functions of polyfit and corrcoef of Numpy to determine respectively the equation of the linear model and the degree of correlation r 2.
3. LEARN :
   At first, you will apply a KNN (K nearest neighbors) for to solve this problem of regression. You can vary the number of neighbors as well as the type of distance used to determine the influence of these hyper- parameters on the performance of this non-parametric model.
4. HYPER-PARAMETERS :
   Similarly, you will apply an SVR (epsilon-SVR or nu-SVR) whose you will vary the value of the hyper-parameters.
5. Through this question, we will try to estimate the amount of data (ie the number of individuals) necessary to obtain quality learning with a SVR. To do this, we will draw learning curves. The idea is to vary the size of the learning set (between [1%, 100%]) and determine the performance of the model obtained on the test dataset, which meanwhile, is unchanged. To answer this question, you will use the configuration that allowed you to to obtain the best results from the previous question

3 Unsupervised learning

3.1 Classification - OneClass SVM

An anomaly detection problem can be solved using binary classifiers. The limit of such approach is related to the nature of anomaly problem wich is variable and unpredictable. Thus it is better to turn towards "one-class" models, i.e. models dedicated to the recognition of a single class.

Question :
In this exercise, you will compare the performance achieved by a OneClass SVM (OCSVM) and Forest Isolation on the following database:

```
randomGenerator = np.random.RandomState(42)
# Training dataset
# Normal data
X = 0.3 * randomGenerator.randn(100, 2)
X_train = np.r_[X + 2, X - 2]
# Validation dataset
# Normal data
X = 0.3 * randomGenerator.randn(20, 2)
X_validation = np.r_[X + 2, X - 2]
# Testing dataset
# Normal data
X = 0.3 * randomGenerator.randn(20, 2)
X_test = np.r_[X + 2, X { 2]
# Anomalies
X_outliers = randomGenerator.uniform(low=-4, high=4, size=(20, 2))
```