Sklearn Tutorial

February 12, 2018

1 Machine Learning with the Scikit-Learn toolkit

1.1 Overview

- Machine learning library written in Python
- Simple and efficient, for both experts and non-experts
- Classical, well-established machine learning algorithms
- Shipped with documentation and examples
- BSD 3 license

1.2 Community driven development

- 20~ core developers (mostly researchers)
- 500+ occasional contributors
- All working publicly together on GitHub
- Emphasis on keeping the project maintainable
- Style consistency
- Unit-test coverage
- Documentation and examples
- Code review

1.3 Python stack for data analysis

- The open source Python ecosystem provides a standalone, versatile and powerful scientific working environment, including: NumPy, SciPy, IPython, Matplotlib, Pandas, and many others...
- Scikit-Learn builds upon NumPy and SciPy and complements this scientific environment with machine learning algorithms;
- By design, Scikit-Learn is non-intrusive, easy to use and easy to combine with other libraries;
- Core algorithms are implemented in low-level languages.

1.4 Algorithms

- Supervised learning:
- Linear models (Ridge, Lasso, Elastic Net, ...)
- Support Vector Machines
- Tree-based methods (Random Forests, Bagging, GBRT, ...)

- Nearest neighbors
- Neural networks
- Gaussian Processes
- Feature selection

1.5 Unsupervised learning:

- Clustering (KMeans, Ward, ...)
- Matrix decomposition (PCA, ICA, ...)
- Density estimation
- Outlier detection

1.6 Model selection and evaluation:

- Cross-validation
- Grid-search
- Lots of metrics

1.7 Framework

Data comes as a finite learning set $\mathcal{L} = (X, y)$ where Input samples are given as an array X of shape n_samples $\times$ n_features, taking their values in $\mathcal{X}$; Output values are given as an array y, taking symbolic values in $\mathcal{Y}$. The goal of supervised classification is to build an estimator $\varphi_{\mathcal{L}} : \mathcal{X} \mapsto \mathcal{Y}$ minimizing

$$Err(\varphi_{\mathcal{L}}) = \mathbb{E}_{X,Y}\{L(Y, \varphi_{\mathcal{L}}(X))\}$$

where L is a loss function, e.g., the zero-one loss for classification $L_{01}(Y, \hat{Y}) = 1(Y \neq \hat{Y})$.

1.8 Applications

- Classifying signal from background events;
- Diagnosing disease from symptoms;
- Recognising cats in pictures;
- Identifying body parts with Kinect cameras;
- ...

1.9 Datasets

Sklearn has a number of datasets that come with the library. We can import them with the python import command.

```
In [1]: from sklearn import datasets
        dir (datasets)

Out[1]: ['__all__',
         '__builtins__',
         '__cached__',
         '__doc__',
         '__file__',
```

```
'__loader__',
'__name__',
'__package__',
'__path__',
'__spec__',
'_svmlight_format',
'base',
'california_housing',
'clear_data_home',
'covtype',
'dump_svmlight_file',
'fetch_20newsgroups',
'fetch_20newsgroups_vectorized',
'fetch_california_housing',
'fetch_covtype',
'fetch_kddcup99',
'fetch_lfw_pairs',
'fetch_lfw_people',
'fetch_mldata',
'fetch_olivetti_faces',
'fetch_rcv1',
'fetch_species_distributions',
'get_data_home',
'kddcup99',
'lfw',
'load_boston',
'load_breast_cancer',
'load_diabetes',
'load_digits',
'load_files',
'load_iris',
'load_linnerud',
'load_mlcomp',
'load_sample_image',
'load_sample_images',
'load_svmlight_file',
'load_svmlight_files',
'load_wine',
'make_biclusters',
'make_blobs',
'make_checkerboard',
'make_circles',
'make_classification',
'make_friedman1',
'make_friedman2',
'make_friedman3',
'make_gaussian_quantiles',
'make_hastie_10_2',
```

```
 'make_low_rank_matrix',
 'make_moons',
 'make_multilabel_classification',
 'make_regression',
 'make_s_curve',
 'make_sparse_coded_signal',
 'make_sparse_spd_matrix',
 'make_sparse_uncorrelated',
 'make_spd_matrix',
 'make_swiss_roll',
 'mlcomp',
 'mldata',
 'mldata_filename',
 'olivetti_faces',
 'rcv1',
 'samples_generator',
 'species_distributions',
 'svmlight_format',
 'twenty_newsgroups']
```

1.10 Our first ML program.

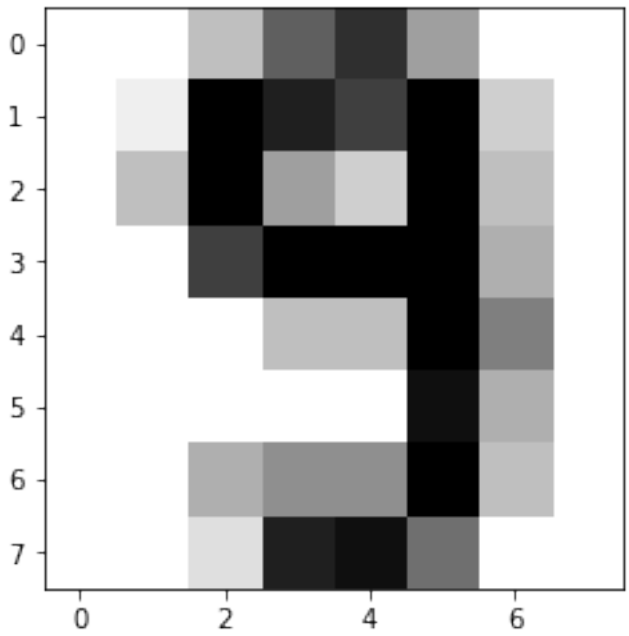
This program is going to use the support vector machine algorithm supplied by SKLearn to classify 8 X 8 greyscale images of numbers. I.e. it is going to try and classify which type of number is represented by that image.

The idea is to show that creating ML algorithms using SKLearn is not difficult.

```
In [4]: # %load ../examples/digits.py
        #!/usr/bin/env python3
        ''' This is our first example of a machine learning algorithm.
            Here we import the digits dataset (A dataset of grayscale images)
            and attempt to classify each image from 0-9.
        '''


        # Import the necessary modules from sklearn and matplotlib
        from sklearn import svm
        from sklearn import datasets
        import matplotlib.pyplot as plt


        # Load the dataset.
        digits = datasets.load_digits()

        #print (digits.data)
        #print (digits.data.shape)

        # We're going to use a support vector machine.  The gamma and C parameters are
```

```
        # known as hyperparameters.  Gamma is the learning rate.
        #

        clf = svm.SVC(gamma=0.0001, C=100)

        # We're going to use 1757 out of the 1767 digits for our training data.  The remainder
        # for our test data.
        x,y = digits.data[:-10],digits.target[:-10]

        # Fit a margin classification for the data.
        clf.fit(x,y)

        # Test the classifier to see how well it predicts the number from the image.
        print ('Prediction: ', clf.predict(digits.data[[-5]]))

        # Plot the image to see if what the classifier predicts matches what we expect.
        plt.imshow(digits.images[-5], cmap=plt.cm.gray_r,interpolation='nearest')
        plt.show()
```

```
Prediction:  [9]
```



In general, a learning problem considers a set of n samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and, for instance, a multi-dimensional entry (aka multivariate data), it is said to have several attributes or features.

We can separate learning problems in a few large categories:

- supervised learning, in which the data comes with additional attributes that we want to predict (Click here to go to the scikit-learn supervised learning page).This problem can be either:

  - classification: samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. An example of classification problem would be the handwritten digit recognition example, in which the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the n samples provided, one is to try to label them with the correct category or class.
  - regression: if the desired output consists of one or more continuous variables, then the task is called regression. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.

- unsupervised learning, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization (Click here to go to the Scikit-Learn unsupervised learning page).

1.11 Training set and testing set

Machine learning is about learning some properties of a data set and applying them to new data. This is why a common practice in machine learning to evaluate an algorithm is to split the data at hand into two sets, one that we call the training set on which we learn data properties and one that we call the testing set on which we test these properties.

1.12 Loading a dataset

scikit-learn comes with a few standard datasets, for instance the iris and digits datasets for classification and the boston house prices dataset for regression.

A dataset is a dictionary-like object that holds all the data and some metadata about the data. This data is stored in the .data member, which is a n_samples, n_features array. In the case of supervised problem, one or more response variables are stored in the .target member. More details on the different datasets can be found in the dedicated section.

For instance, in the case of the digits dataset, digits.data gives access to the features that can be used to classify the digits samples:

```
In [7]: from sklearn import datasets
        iris = datasets.load_iris()
        digits = datasets.load_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.]]
```

and digits.target gives the ground truth for the digit dataset, that is the number corresponding to each digit image that we are trying to learn:

```
In [8]: print (digits.target)
```

```
[0 1 2 ..., 8 9 8]
```

1.13 Shape of the data arrays

The data is always a 2D array, shape (n_samples, n_features), although the original data may have had a different shape. In the case of the digits, each original sample is an image of shape (8, 8) and can be accessed using:

```
In [9]: print (digits.images[0])
```

```
[[  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.]]
```

1.14 Loading from external datasets

scikit-learn works on any numeric data stored as numpy arrays or scipy sparse matrices. Other types that are convertible to numeric arrays such as pandas DataFrame are also acceptable.

Here are some recommended ways to load standard columnar data into a format usable by scikit-learn: - pandas.io provides tools to read data from common formats including CSV, Excel, JSON and SQL. DataFrames may also be constructed from lists of tuples or dicts. Pandas handles heterogeneous data smoothly and provides tools for manipulation and conversion into a numeric array suitable for scikit-learn. - scipy.io specializes in binary formats often used in scientific computing context such as .mat and .arff - numpy/routines.io for standard loading of columnar data into numpy arrays - scikit-learn's datasets.load_svmlight_file for the svmlight or libSVM sparse format - scikit-learn's datasets.load_files for directories of text files where the name of each directory is the name of each category and each file inside of each directory corresponds to one sample from that category

For some miscellaneous data such as images, videos, and audio, you may wish to refer to: - skimage.io or Imageio for loading images and videos to numpy arrays - scipy.misc.imread (requires the Pillow package) to load pixel intensities data from various image file formats - scipy.io.wavfile.read for reading WAV files into a numpy array