Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow

Concepts, Tools, and Techniques to Build Intelligent Systems

Stephen CUI[1]

October 30, 2022

[1]cuixuanStephen@gmail.com

Contents

Part I

The Fundamentals of Machine Learning

Chapter 1

Classification

Now we will turn our attention to classification systems.

1.1 MNIST

In this chapter we will be using the MNIST dataset, which is a set of 70,000 small images of digits handwritten by high school students and employees of the US Census Bureau. Each image is labeled with the digit it represents. This set has been studied so much that it is often called the "hello world" of machine learning: whenever people come up with a new classification algorithm they are curious to see how it will perform on MNIST, and anyone who learns machine learning tackles this dataset sooner or later.

Scikit-Learn provides many helper functions to download popular datasets. The following code fetches the MNIST dataset from OpenML.org:

```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', as_frame=False)
```

The sklearn.datasets package contains mostly three types of functions:

- `fetch_*` functions such as `fetch_openml()` to download real-life datasets;
- `load_*` functions to load small toy datasets bundled with Scikit-Learn (so they don't need to be downloaded over the internet);
- `make_*` functions to generate fake datasets, useful for tests;

Generated datasets are usually returned as an (X, y) tuple containing the input data and the targets, both as NumPy arrays. Other datasets are returned as sklearn.utils.Bunch objects, which are dictionaries whose entries can also be accessed as attributes. They generally contain the following entries:

- `DESCR`: A description of the dataset
- `data`: The input data, usually as a 2D NumPy array

- `target`: The labels, usually as a 1D NumPy array

The `fetch_openml()` function is a bit unusual since by default it returns the inputs as a Pandas DataFrame and the labels as a Pandas Series (unless the dataset is sparse). But the MNIST dataset contains images, and DataFrames aren't ideal for that, so it's preferable to set `as_frame=False` to get the data as NumPy arrays instead. Let's look at these arrays:

```
X, y = mnist.data, mnist.target
X.shape, y.shape
```

There are 70,000 images, and each image has 784 features. This is because each image is 28×28 pixels, and each feature simply represents one pixel's intensity, from 0 (white) to 255 (black). Let's take a peek at one digit from the dataset (Figure 1.1).

```
import matplotlib.pyplot as plt

def plot_digit(image_data):
    image = image_data.reshape(28, 28)
    plt.imshow(image, cmap='binary')
    plt.axis('off')

some_digit = X[0]
plot_digit(some_digit)
```

To give you a feel for the complexity of the classification task, Figure 1.2 shows a few more images from the MNIST dataset.

在你进一步了解数据的时候，你应该将数据切分为训练集和测试集。The MNIST dataset returned by `fetch_openml()` is actually already split into a training set (the first 60,000 images) and a test set (the last 10,000 images)[1]:

```
X_train, X_test, y_train, y_test = X[:60_000], X[60_000:], y[:60_000], y[60_000:]
```

[1] Datasets returned by `fetch_openml()` are not always shuffled or split.

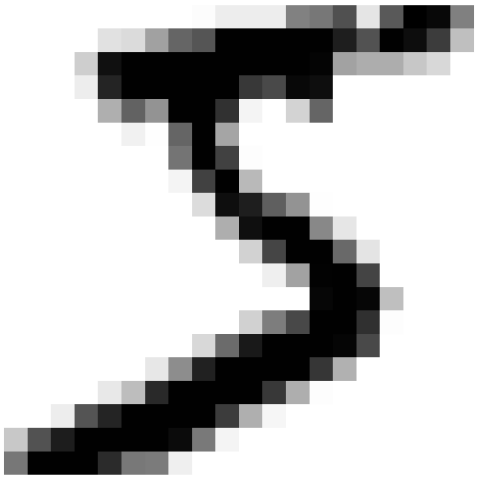

Figure 1.1: Example of an MNIST image

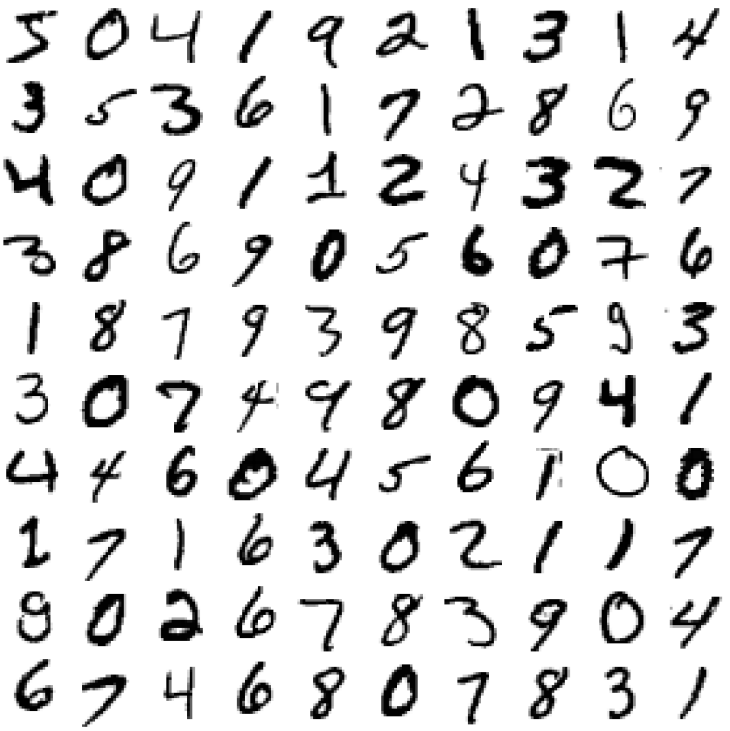

Figure 1.2: Digits from the MNIST dataset

1.2 Training a Binary Classifier

1.3 Performance Measures

1.3.1 Measuring Accuracy Using Cross-Validation

1.3.2 Confusion Matrices

1.3.3 Precision and Recall

1.3.4 The Precision/Recall Trade-off

1.3.5 The ROC Curve

1.4 Multiclass Classification

1.4.1

1.4.2

1.5

1.5.1

1.5.2

1.6

1.7

1.8

Chapter 2

Decision Trees

Chapter 3

Ensemble Learning and Random Forests

Part II

Neural Networks and Deep Learning

Chapter 4

Training and Deploying TensorFlow Models at Scale