第3章

Python机器学习工具箱

用户自行实现机器学习的各种经典算法是好的，且好处很明显，能让自己对机器学习算法的细节了然于胸。但即使使用了简单、高效的 Python 来编写代码，实现起来代码量依然很大，更何况所写的代码可能并不专业，对很多意外情况可能考虑不足，算法的性能难以保证。这时用户可能会有这样的需求：是否有成熟的机器学习框架，能让我们更加关注算法的业务逻辑，而不必事无巨细地重造“轮子”呢？如求方差、求均方根误差这类通用函数可否不必自己编写？事实上对于一些经典的机器学习算法，一些专家或工程师早已将其实现，大概率上，由他们实现的算法在各种条件下的完备性及数值计算的稳定性都要远胜于用户自己实现的算法。在机器学习领域，scikit-learn（简称 sklearn）就是由专业人士开发的久经考验的机器学习框架。另外，OpenAI Gym 是专门用于强化学习算法的工具包。

本章将介绍 scikit-learn 和 OpenAI Gym 的基础知识和基本使用方法，如功能、原理、安装和基本使用方法，其详细的使用方法将在本书的 Python 实践部分进行介绍。

3.1 机器学习的利器——scikit-learn

3.1.1 scikit-learn 的基础知识

scikit-learn 是 Python 的一个开源机器学习模块，它建立在 NumPy、SciPy 和 Matplotlib 模块之上，能够为用户提供各种机器学习算法接口。scikit-learn 最大的特点就是为用户提供各种机器学习算法接口，可以让用户简单、高效地进行数据挖掘和数据分析。

scikit-learn 包含众多顶级机器学习算法，主要有六大基本功能，分别是分类、回归、聚类、降维、模型选择和预处理。scikit-learn 拥有非常活跃的用户社区，基本上其所有的

功能都有非常详尽的文档供用户查阅。大家可以研读 scikit-learn 的用户指南及文档，以便对其算法的使用有更充分的了解。其界面如图 3.1 所示，网址为 https://scikit-learn.org/dev/sklearn。

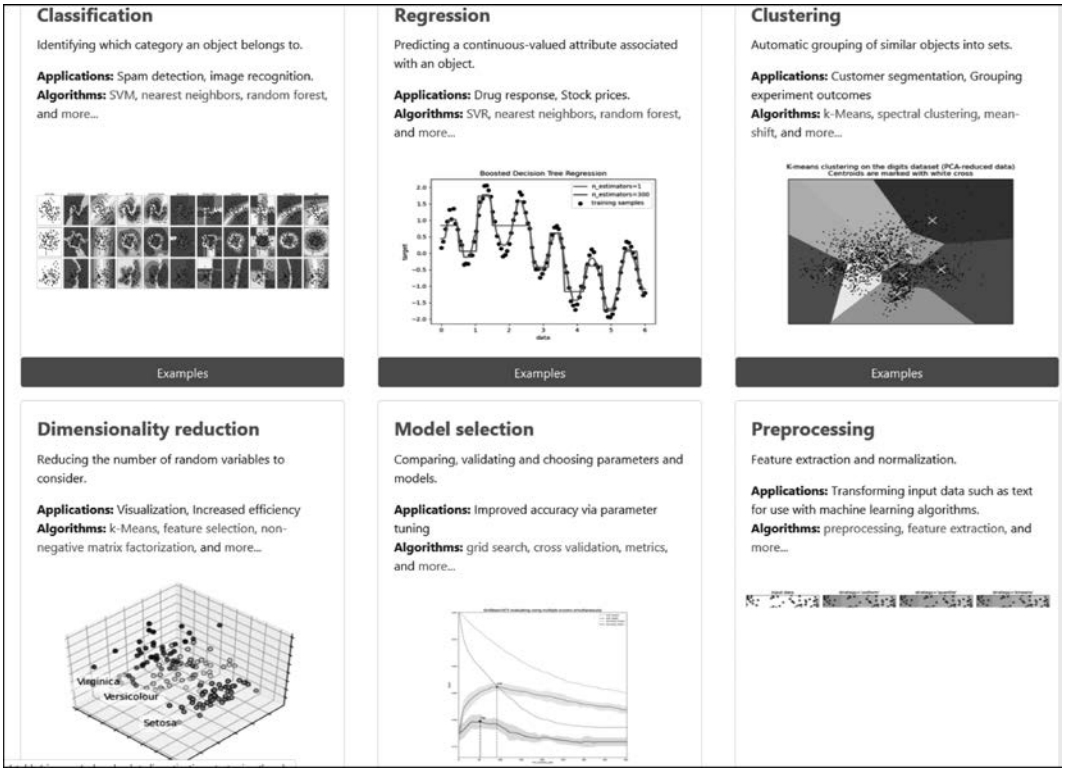

图 3.1　scikit-learn 界面

1. 分类（Classification）

识别某个对象属于哪个类别，常用的算法有 SVM（支持向量机）、nearest neighbors（最近邻）、random forest（随机森林），常见的应用有垃圾邮件识别、图像识别等。引入 scikit-learn 分类函数的代码如下：

```
from sklearn import SomeClassifier
from sklearn.linear_model import SomeClassifier
from sklearn.ensemble import SomeClassifier
```

2. 回归（Regression）

预测与对象相关联的连续值属性，常见的算法有 SVR（支持向量机）、ridge regression（岭回归）、Lasso，常见的应用有药物反应、预测股价。引入 scikit-learn 回归函数的代码如下：

```
from sklearn import SomeRegressor
from sklearn.linear_model import SomeRegressor
from sklearn.ensemble import SomeRegressor
```

3. 聚类(Clustering)

将相似对象自动分组,常用的算法有 k 均值、spectral clustering、mean-shift,常见的应用有客户细分、分组实验结果。引入 scikit-learn 聚类函数的代码如下:

```
from sklearn.cluster import SomeModel
```

4. 降维(Dimensionality reduction)

减少要考虑的随机变量的数量,常见的算法有 PCA(主成分分析)、feature selection(特征选择)、non-negative matrix factorization(非负矩阵分解),常见的应用有可视化、提高效率。引入 scikit-learn 降维函数的代码如下:

```
from sklearn.decomposition import SomeModel
```

5. 模型选择(Model selection)

比较,验证,选择参数和模型,常用的模块有 grid search(网格搜索)、cross validation(交叉验证)、metrics(度量)。它的目标是通过参数调整提高精度。引入 scikit-learn 模块选择函数的代码如下:

```
from sklearn.model_selection import SomeModel
```

6. 预处理(Preprocessing)

特征提取和归一化,常用的模块有 preprocessing、feature extraction,常见的应用是把输入数据(例如文本)转换为机器学习算法可用的数据。引入 scikit-learn 预处理函数的代码如下:

```
from sklearn.preprocessing import SomeModel
```

SomeClassifier、SomeRegressor、SomeModel 其实都叫作估计器(estimator),就像 Python 中"万物皆对象"那样,sklearn 中"万物皆估计器"。

7. 数据集(Dataset)

此外,sklearn 中还自带有很多数据集,引入它们的代码如下:

```
from sklearn.datasets import SomeData
```

3.1.2　scikit-learn 的安装

在安装 scikit-learn 之前需要安装以下依赖条件：

(1) Python，2.6 或 3.3 以上版本。

(2) NumPy，1.6.1 以上版本。

(3) SciPy，0.9 以上版本。

在 Linux 系统下，可以直接打开 terminal，使用 pip 进行安装，指令如下：

```
sudo pip install -U scikit-learn
```

在 PyCharm 中选择 File→Settings 命令，在打开的对话框中选择 Project Interpreter 选项，然后单击加号和 OK 按钮，如图 3.2 所示。

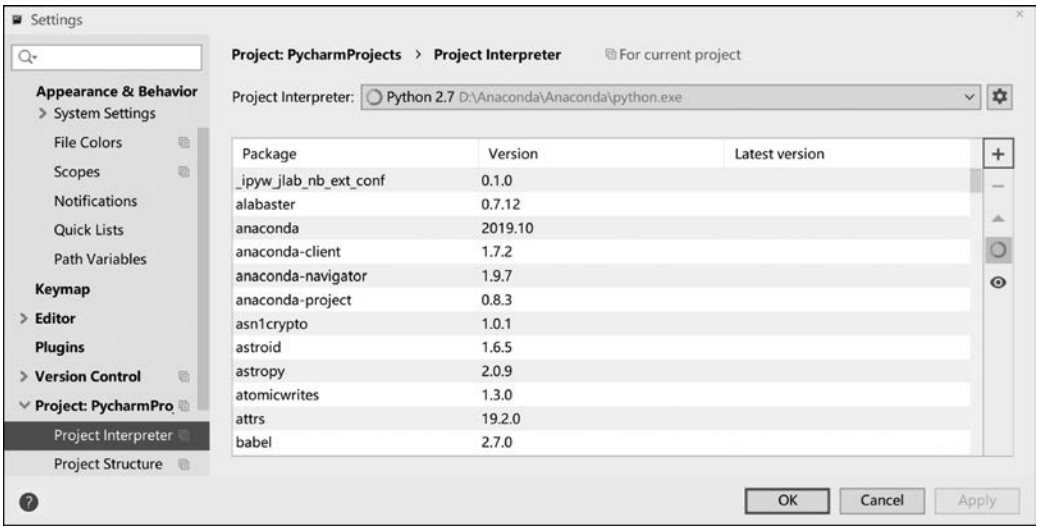

图 3.2　安装步骤图

在 Available Packages 对话框中搜索 scikit-learn，然后选择搜索到的 scikit-learn 版本，单击 Install Package 按钮进行安装，如图 3.3 所示。

安装后进行测试，在 Python 文件中输入以下代码：

```
from sklearn import preprocessing
```

或者 import 其他功能模块，如果程序不报错，则说明安装成功。

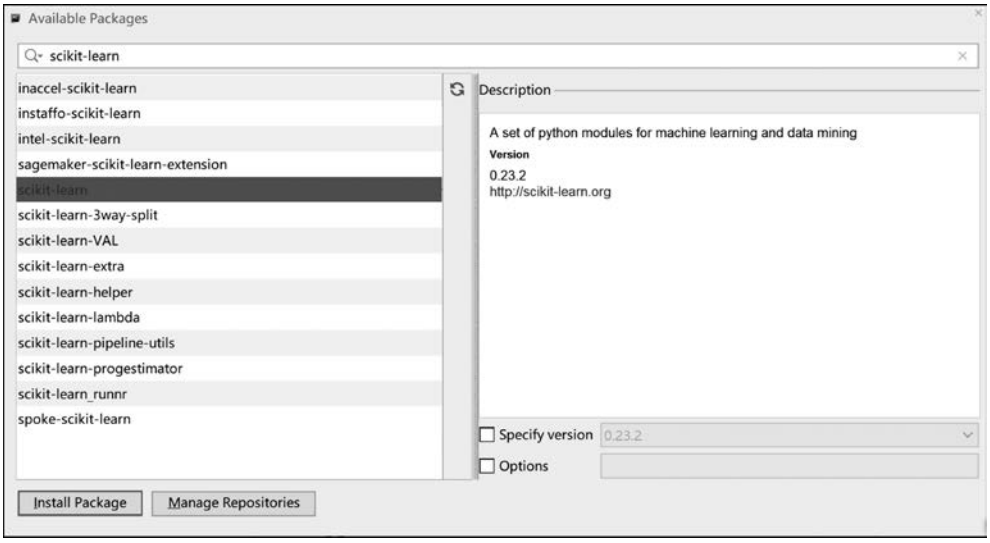

图 3.3　选择 scikit-learn 版本

3.1.3　基本功能的介绍

1. 载入数据

scikit-learn 内包含了常用的机器学习数据集，例如做分类的 iris 和 digit 数据集，用于回归的经典数据集 Boston house prices 等。导入 iris 数据集的代码如下：

```
from sklearn import datasets
iris = datasets.load_iris()
```

scikit-learn 载入的数据集是以类似于字典的形式存放的，该对象中包含了所有有关该数据的信息。其中的数据值统一存放在.data 的成员中，例如要将 iris 数据显示出来，只需显示 iris 的 data 成员即可，代码如下：

```
print(iris.data)
```

数据都是以 n 维（n 个特征）矩阵形式存放和展现的，iris 数据中每个实例有 4 维特征，分别为 sepal length、sepal width、petal length 和 petal width。显示的 iris 数据如下：

```
[[ 5.1  3.5  1.4  0.2]
 [ 4.9  3   1.4  0.2]
       ...
 [ 5.9  3   5.1  1.8]]
```

对于监督学习，比如分类问题，数据中会包含对应的分类结果，其存放在.target 成员中，代码如下：

```
print(iris.target)
```

对于 iris 数据而言,就是各个实例的分类结果:

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

接下来按照 6 个步骤将上面所得到的数据进行训练,进行一次最基础的机器学习。

(1) 加载训练模型所用的数据集。

(2) 采用合适的比例将数据集划分为训练集和测试集。

(3) 选取或者创建合适的训练模型。

(4) 将训练集中的数据输入模型中进行训练。

(5) 通过第(4)步的训练大致确定模型所用的合理参数。

(6) 将测试集中的数据输入模型中,将模型得到的结果和真实的结果进行比较,再次调整参数。

示例如下:

```
from sklearn import datasets
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
iris = datasets.load_iris()
X = iris.data
y = iris.target
#新建一个模型(参数默认)
iris_model = LinearRegression()
#分割训练集、测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3., random_state=7)
#训练该模型
iris_model.fit(X_train,y_train)
#返回模型参数列表
print(iris_model.get_params())
#模型在训练集上的评分
print(iris_model.score(X_train, y_train))
#模型在测试集上的评分
print(iris_model.score(X_test, y_test))
#使用模型进行预测
y_pred = iris_model.predict(X_test)
print('预测标签:', y_pred[:3])
print('真实标签:', y_test[:3])
```

最终得到的结果如下:

```
{'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': False}
0.9451696710635755
```

```
0.8959790715079212
预测标签: [ 1.66096074  1.39389456 -0.02571618]
真实标签: [2 1 0]
```

2. 学习与预测

scikit-learn 提供了各种机器学习算法的接口，允许用户很方便地使用。每个算法的调用就像一个黑箱，对于用户来说，只需要根据自己的需求设置相应的参数即可。

例如，调用最常用的支撑向量分类机(SVM)：

```
from sklearn import svm
clf = svm.SVC(gamma = 0.001, C = 100.)
print(clf)
```

输出为分类器的具体参数信息：

```
SVC(C = 100.0, cache_size = 200, class_weight = None, coef0 = 0.0,
decision_function_shape = 'ovr', degree = 3, gamma = 0.001, kernel = 'rbf',
  max_iter = -1, probability = False, random_state = None, shrinking = True,
  tol = 0.001, verbose = False)
```

分类器的学习和预测可以分别使用 fit(X,Y)和 predict(T)来实现。将 digit 数据划分为训练集和测试集，前 $n-1$ 个实例为训练集，最后一个为测试集。然后使用 fit 和 predict 分别完成学习和预测，代码如下：

```
from sklearn import datasets
from sklearn import svm
clf = svm.SVC(gamma = 0.001, C = 100.)
digits = datasets.load_digits()
clf.fit(digits.data[:-1], digits.target[:-1])
result = clf.predict(digits.data[9:10])
print(result)
```

输出结果为：

```
[9]
```

通过上述例子，简单地介绍了如何使用 scikit-learn 解决分类问题，实际上这个问题要复杂得多，更复杂、更具体的问题将在 Python 实践部分进行讲解。

3.2 强化学习的利器——OpenAI Gym

1. OpenAI Gym 简介

OpenAI Gym 是一款用于研发和比较强化学习算法的工具包，它支持训练智能体(agent)做任何事——从行走到围棋之类的游戏都在该范围中。它与其他的数值计算库

兼容，例如 tensorflow 或者 theano 库。现在主要支持的是 Python 语言。

OpenAI Gym 官方提供的文档的网址为 https://gym.openai.com/docs/，读者可以访问官方网站对 OpenAI Gym 进行更详细的学习。

2. OpenAI Gym 的组成

OpenAI Gym 包含以下两个部分。

(1) gym 开源：包含一个测试问题集，每个问题称为一个环境(environment)，可以用于用户的强化学习算法开发，这些环境有共享的接口，允许用户设计通用的算法，如 Atari、CartPole 等。

(2) OpenAI Gym 服务：提供一个站点和 api，允许用户对他们训练的算法进行性能比较。

3. 强化学习与 OpenAI Gym

强化学习(reinforcement learning，RL)是机器学习的一个分支，考虑的是做出一系列决策。它假定有一个智能体存在于环境中。在每一步中，智能体采取一个行动，随后从环境中收到观察与回报。RL 算法寻求的是，在一个原先毫无了解的环境中通过一段学习过程(通常包括许多试错)让智能体收到的总体回报最大化。

在强化学习中有两个基本概念，一个是环境(environment)，称为外部世界，另一个是智能体 agent(写的算法)。agent 发送 action 至 environment，environment 返回观察和回报。OpenAI Gym 的核心接口是 Env，作为统一的环境接口。Env 包含以下核心方法。

(1) env.reset(self)：重置环境的状态，返回观察。

(2) env.step(self，action)：推进一个时间步长，返回 observation、reward、done、info。

(3) env.render(self，mode='human'，close=False)：重绘环境的一帧。默认模式一般比较友好，例如弹出一个窗口。

4. OpenAI Gym 的安装

(1) 安装 OpenAI Gym 所需要的所有依赖包。

```
$ apt-get install -y python-numpy python-dev cmake zlib1g-dev libjpeg-dev xvfb libav
-tools xorg-dev python-opengl libboost-all-dev libsdl2-dev swig
```

(2) 通过 git 方式安装 OpenAI Gym。

```
$ git clone https://github.com/openai/gym   #下载安装包
$ cd gym                                    #进入 gym 文件夹
$ pip install -e .                          #最小安装方式
or
$ pip install -e .[all]                     #全安装方式(需要 cmake 和 pip)
```

(3) 通过 pip 方式安装。

```
$ pip install gym                          #最小安装方式
or
$ pip install gym[all]                     #全安装方式(将 gym 视为一个安装包获取)
```

其中,步骤(2)和步骤(3)都是按照 OpenAI Gym 的方式,可任选一个,推荐选步骤(3)。

5. OpenAI Gym 的 demo 运行

为了让读者快速了解 OpenAI Gym 的操作,这里参考官方文档编写一个 demo,体验一下 OpenAI Gym 平台。以倒立摆(CartPole)为例,在工作目录中建立一个 Python 文件,将文件命名为 CartPole.py,代码如下:

```
import gym                                 #导入 OpenAI Gym 函数
env = gym.make('CartPole-v0')              #建立一个 gym 环境
for i_episode in range(20):
    observation = env.reset()              #重置环境的状态
    for t in range(100):
        env.render()                       #重绘环境的一帧
        print(observation)                 #打印观察值
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break
env.close()
```

运行 Python 函数:

```
$ python CartPole.py
```

显示一段倒立摆训练的视频,视频截图如图 3.4 所示:

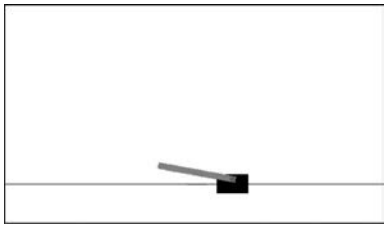

图 3.4 倒立摆视频

并输出观察值:

```
[-0.061586   -0.75893141  0.05793238  1.15547541]
[-0.07676463 -0.95475889  0.08104189  1.46574644]
[-0.0958598  -1.15077434  0.11035682  1.78260485]
```

```
[ -0.11887529 -0.95705275  0.14600892  1.5261692 ]
[ -0.13801635 -0.7639636   0.1765323   1.28239155]
[ -0.15329562 -0.57147373  0.20218013  1.04977545]
Episode finished after 14 timesteps
[ -0.02786724  0.00361763 -0.03938967 -0.01611184]
[ -0.02779488 -0.19091794 -0.03971191  0.26388759]
[ -0.03161324  0.00474768 -0.03443415 -0.04105167]
```

通过上述例子，简单地介绍了如何使用 OpenAI Gym 平台来做强化学习，实际上这个问题要复杂得多，更复杂、更具体的问题将在 Python 实践部分进行讲解。

本章参考文献

[1] https://scikit-learn.org/stable/.

[2] https://segmentfault.com/a/1190000013765279.

[3] https://gym.openai.com/docs/.