# sklearn 模型的保存与加载 - 腾讯云开发者社区-腾讯云

## sklearn 模型的保存与加载

> 🔗 原文链接： https://cloud.tencent.com/developer...

在我们基于训练集训练了 `sklearn` 模型之后，常常需要将预测的模型保存到文件中，然后将其还原，以便在新的数据集上测试模型或比较不同模型的性能。其实把模型导出的这个过程也称为「对象序列化」-- 将对象转换为可通过网络传输或可以存储到本地磁盘的数据格式，而还原的过程称为「反序列化」。

本文将介绍实现这个过程的三种方法，每种方法都有其优缺点：

1.Pickle[1]， 这是用于对象序列化的标准 Python 工具。2.Joblib[2] 库，它可以对包含大型数据数组的对象轻松进行序列化和反序列化。3.手动编写函数将对象保存为 JSON[3]，并从 JSON 格式载入模型。

这些方法都不代表最佳的解决方案，我们应根据项目需求选择合适的方法。

### 建立模型

首先，让我们需要创建模型。在示例中，我们将使用 Logistic回归[4] 模型和 Iris数据集[5]。让我们导入所需的库，加载数据，并将其拆分为训练集和测试集。

```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
# Load and split data
data = load_iris()
Xtrain, Xtest, Ytrain, Ytest = train_test_split(data.data, data.target, test_siz
```

接下来，使用一些非默认参数创建模型并将其拟合训练数据。

```
# Create a model
model = LogisticRegression(C=0.1,
                           max_iter=20,
                           fit_intercept=True,
                           n_jobs=3,
                           solver='liblinear')
```

```
model.fit(Xtrain, Ytrain)
```

最终得到的模型：

```
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=True,
    intercept_scaling=1, max_iter=20, multi_class='ovr', n_jobs=3,
    penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
    verbose=0, warm_start=False)
```

## 使用 `Pickle` 模块

在下面的几行代码中，我们会把上面得到的模型保存到 `pickle_model.pkl` 文件中，然后将其载入。最后，使用载入的模型基于测试数据计算 Accuracy，并输出预测结果。

```
import pickle
#
# Create your model here (same as above)
#
# Save to file in the current working directory
pkl_filename = "pickle_model.pkl"
with open(pkl_filename, 'wb') as file:
    pickle.dump(model, file)
# Load from file
with open(pkl_filename, 'rb') as file:
    pickle_model = pickle.load(file)
# Calculate the accuracy score and predict target values
score = pickle_model.score(Xtest, Ytest)
print("Test score: {0:.2f} %".format(100 * score))
Ypredict = pickle_model.predict(Xtest)
```

我们也可以将一些过程中的参数用 tuple 形式保存下来：

```
tuple_objects = (model, Xtrain, Ytrain, score)
# Save tuple
pickle.dump(tuple_objects, open("tuple_model.pkl", 'wb'))
# Restore tuple
pickled_model, pickled_Xtrain, pickled_Ytrain, pickled_score = pickle.load(open(
```

> `cPickle` 是用 C 编码的 `pickle` 模块，性能更好，推荐在大多数的场景中使用该模块。

## 使用 `Joblib` 模块

`joblib` 是 `sklearn` 中自带的一个工具。在多数场景下， `joblib` 的性能要优于 `pickle` ，尤其是当数据量较大的情况更加明显。

```
from sklearn.externals import joblib
# Save to file in the current working directory
joblib_file = "joblib_model.pkl"
joblib.dump(model, joblib_file)
# Load from file
joblib_model = joblib.load(joblib_file)
# Calculate the accuracy and predictions
score = joblib_model.score(Xtest, Ytest)
print("Test score: {0:.2f} %".format(100 * score))
Ypredict = pickle_model.predict(Xtest)
```

从示例中可以看出，与 `Pickle` 相比， `Joblib` 库提供了更简单的工作流程。 `Pickle` 要求将文件对象作为参数传递，而 `Joblib` 可以同时处理文件对象和字符串文件名。如果您的模型包含大型数组，则每个数组将存储在一个单独的文件中，但是保存和还原过程将保持不变。 `Joblib` 还允许使用不同的压缩方法，例如 `zlib` ， `gzip` ， `bz2` 等。

## 用 JSON 保存和还原模型

在项目过程中，很多时候并不适合用 `Pickle` 或 `Joblib` 模型，比如会遇到一些兼容性问题。下面的示例展示了如何用 JSON 手动保存和还原对象。这种方法也更加灵活，我们可以自己选择需要保存的数据，比如模型的参数，权重系数，训练数据等等。为了简化示例，这里我们将仅保存三个参数和训练数据。

```
import json
import numpy as np
class MyLogReg(LogisticRegression):
    # Override the class constructor
    def __init__(self, C=1.0, solver='liblinear', max_iter=100, X_train=None, Y_
        LogisticRegression.__init__(self, C=C, solver=solver, max_iter=max_iter)
        self.X_train = X_train
        self.Y_train = Y_train
    # A method for saving object data to JSON file
    def save_json(self, filepath):
        dict_ = {}
        dict_['C'] = self.C
        dict_['max_iter'] = self.max_iter
        dict_['solver'] = self.solver
        dict_['X_train'] = self.X_train.tolist() if self.X_train is not None els
```

```
        dict_['Y_train'] = self.Y_train.tolist() if self.Y_train is not None els
        # Creat json and save to file
        json_txt = json.dumps(dict_, indent=4)
        with open(filepath, 'w') as file:
            file.write(json_txt)
    # A method for loading data from JSON file
    def load_json(self, filepath):
        with open(filepath, 'r') as file:
            dict_ = json.load(file)
        self.C = dict_['C']
        self.max_iter = dict_['max_iter']
        self.solver = dict_['solver']
        self.X_train = np.asarray(dict_['X_train']) if dict_['X_train'] != 'None
        self.Y_train = np.asarray(dict_['Y_train']) if dict_['Y_train'] != 'None
```

下面我们就测试一下 `MyLogReg` 函数。首先，创建一个对象 `mylogreg` ，将训练数据传递给它，然后将其保存到文件中。然后，创建一个新对象 `json_mylogreg` 并调用 `load_json` 方法从文件中加载数据。

```
filepath = "mylogreg.json"
# Create a model and train it
mylogreg = MyLogReg(X_train=Xtrain, Y_train=Ytrain)
mylogreg.save_json(filepath)
# Create a new object and load its data from JSON file
json_mylogreg = MyLogReg()
json_mylogreg.load_json(filepath)
json_mylogreg
```

输入结果如下，我们可以查看参数和训练数据。

```
MyLogReg(C=1.0,
     X_train=array([[ 4.3,  3. ,  1.1,  0.1],
       [ 5.7,  4.4,  1.5,  0.4],
       ...,
       [ 7.2,  3. ,  5.8,  1.6],
       [ 7.7,  2.8,  6.7,  2. ]]),
     Y_train=array([0, 0, ..., 2, 2]), class_weight=None, dual=False,
     fit_intercept=True, intercept_scaling=1, max_iter=100,
     multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
     solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

使用 JSON 进行数据序列化实际上是将对象保存为字符串格式，所以我们可以用文本编辑器打开和修改 `mylogreg.json` 文件。尽管这种方法对开发人员来说很方便，但其他人员也可以随意查看和修改 JSON 文件的内容，因此安全性较低。而且，这种方法更适用于实例变量较少的对象，例如 `sklearn` 模型，因为任何新变量的添加都需要更改保存和载入的方法。

## 兼容性问题

`Pickle` 和 `Joblib` 的最大缺点就是其兼容性问题，可能与不同模型或 Python 版本有关。

· *Python 版本兼容性*：两种工具的文档都指出，不建议在不同的 Python 版本之间对对象进行序列化以及反序列化。 · *模型兼容性*：在使用 `Pickle` 和 `Joblib` 保存和重新加载的过程中，模型的内部结构应保持不变。

`Pickle` 和 `Joblib` 的最后一个问题与安全性有关。这两个工具都可能包含恶意代码，因此不建议从不受信任或未经 身份验证 的来源加载数据。

## 结论

本文我们描述了用于保存和加载 `sklearn` 模型的三种方法。 `Pickle` 和 `Joblib` 库简单快捷，易于使用，但是在不同的 Python 版本之间存在兼容性问题，且不同模型也有所不同。另一方面，手动编写函数的方法相对来说更为困难，并且需要根据模型结构进行修改，但好处在于，它可以轻松地适应各种需求，也不存在任何兼容性问题。

> 本文翻译整理自：https://stackabuse.com/scikit-learn-save-and-restore-models/

### 引用链接

`[1]` Pickle: *https://docs.python.org/3/library/pickle.html* `[2]` Joblib: *https://pythonhosted.org/joblib/* `[3]` JSON: *https://en.wikipedia.org/wiki/JSON* `[4]` Logistic 回归: *https://en.wikipedia.org/wiki/Logistic_regression* `[5]` Iris数据集: *https://en.wikipedia.org/wiki/Iris_flower_data_set*