**Pytorch数据加载-高维数据加载**

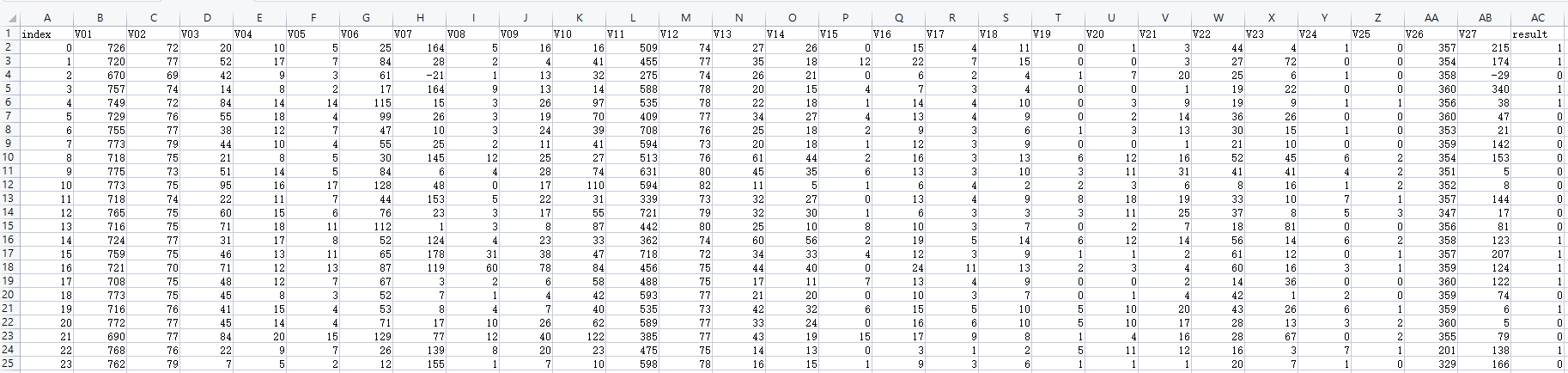
在上次的文章中我们讲述了Python数据加载torch.utils.data中函数和类，详情可以从以下文章中进行查看：

**Pytorch数据加载-torch.utils.data函数和类介绍**

本次文章我们简单的学习一下怎么进行高维数据的加载(如CSV文件)，考虑到csv文件中的数据大多用于分类和回归任务的，因此我们从加载分类和回归任务的数据集进行讲述。

1. **CSV数据加载**

首先我们看一下一个csv文件是怎样进行加载和使用的，我们先来看一个代码，数据形式大致如下：



使用的代码我们如下：

class MyDataset(Dataset):  
  
 def \_\_init\_\_(self, filepath):  
 *# 获取数据* trainData = pd.read\_csv(filepath)  
 *# 转换为Tensor的形式* self.feature = torch.from\_numpy(trainData.iloc[:,1:-1].values)  
 self.target = torch.from\_numpy(trainData.iloc[:,[-1]].values)  
 *# 获取数据的大小* self.len = trainData.shape[0]  
  
 def \_\_getitem\_\_(self, index):  
 *# 获取第index条数据的特征及标签* return self.feature[index], self.target[index]  
  
 def \_\_len\_\_(self):  
 *# 数据长度* return self.len

if \_\_name\_\_ == "\_\_main\_\_":  
 dataset = MyDataset('mydata.csv')  
 print(len(dataset))  
 print(dataset[15])

# 输出如下：

3000

(tensor([670, 69, 42, 9, 3, 61, -21, 1, 13, 32, 275, 74, 26, 21,

0, 6, 2, 4, 1, 7, 20, 25, 6, 1, 0, 358, -29]), tensor([0]))

上述代码中我们继承了Dataset类，并重写了两个方法，注意这里len方法的结果是3000，指的是数据样例个数，也就是mydata.csv中数据的行数，这样我们就完成了csv数据的读取。

1. **回归数据加载**

回归数据也可能存在csv文件中，如果是这样的话大家可以参考上面的例子。这里我们加载sklearn中的房价数据集，代码如下：

import torch  
import numpy as np  
import torch.utils.data as Data  
from sklearn.datasets import load\_boston  
  
import warnings  
warnings.filterwarnings("ignore")  
  
X,Y = load\_boston(return\_X\_y=True)  
print(X.dtype)  *# float64*  
print(Y.dtype) *# float64*  
*# 将数组转换为Tensor*train\_x = torch.from\_numpy(X.astype(np.float32))  
train\_y = torch.from\_numpy(Y.astype(np.float32))  
train\_data = Data.TensorDataset(train\_x,train\_y)  
*# 定义数据加载器*Loader = Data.DataLoader(  
 dataset=train\_data,  
 batch\_size=32,  
 shuffle=True  
)  
  
for inx,(x,y) in enumerate(Loader):  
 print('inx:',inx)  
 print('x:',x)  
 print('y:',y)  
 break

输出这里就不在贴出来了，上述代码的主要思路就是：

1. 将load\_boston()获取的float64数据转换为float32类型
2. 接着使用from\_numpy将数据转换为Tensor张量数据
3. 然后构造数据加载器，设置batch\_size=32即32个样本为一个batch
4. 最后使用for循环获取一次数据
5. **分类数据加载**

分类数据也是我们在建模的时候常见的数据，只不过分类数据集的标签lebal是离散的而已，我们来看一下：

import torch  
import numpy as np  
import torch.utils.data as Data  
from sklearn.datasets import load\_iris  
  
import warnings  
warnings.filterwarnings("ignore")  
  
X,Y = load\_iris(return\_X\_y=True)  
print(X.dtype) *# float64*print(Y.dtype) *# int32  
  
# 将数组转换为Tensor*train\_x = torch.from\_numpy(X.astype(np.float32))  
train\_y = torch.from\_numpy(Y.astype(np.int64))  
train\_data = Data.TensorDataset(train\_x,train\_y)  
*# 定义数据加载器*Loader = Data.DataLoader(  
 dataset=train\_data,  
 batch\_size=32,  
 shuffle=True  
)  
  
for inx,(x,y) in enumerate(Loader):  
 print('inx:',inx)  
 print('x:',x)  
 print('y:',y)  
 break

跟上述的回归数据的加载是一致的，这里就不在赘述了。

1. **参考文档**

<https://pytorch.org/docs/stable/data.html>

https://github.com/lyhue1991/eat\_pytorch\_in\_20\_days