# pytorch 中数据操作和预处理

# 书中的版本有两个类型，一种是CPU,PYTORCH 1.3 , PYTHON 3.6 ,MacOS 系统，

# 另一个是 GPU ,PYTORCH 1.0 CUDA 8.0 , PYTHON 3.6 CentOS Linux 7 (Core) 系统

import torch

import numpy as np

import torch.utils.data as Data

import pandas as pd

from sklearn.datasets import load\_iris

# from sklearn.datasets import load\_boston

# 针对回归，换成另外一个回归数据集

from sklearn.datasets import load\_diabetes

diabetes = load\_diabetes()

X = diabetes.data # 特征数据

Y = diabetes.target # 目标数据

# 查看特征数据和目标数据的形状

print("特征数据形状:", X.shape)

print("目标数据形状:", Y.shape)

# 特征数据形状: (442, 10)

# 目标数据形状: (442,)

print("X:", type(X), X.dtype)

print("Y:", type(Y), Y.dtype)

# X: <class 'numpy.ndarray'> float64

# Y: <class 'numpy.ndarray'> float64

# 先变32 再变成 tensor

train\_xt = torch.from\_numpy(X.astype(np.float32))

train\_yt = torch.from\_numpy(Y.astype(np.float32))

print("train\_xt:", type(train\_xt), train\_xt.dtype)

print("train\_yt:", type(train\_yt), train\_yt.dtype)

# train\_xt: <class 'torch.Tensor'> torch.float32

# train\_yt: <class 'torch.Tensor'> torch.float32

train\_data = Data.TensorDataset(train\_xt, train\_yt)

train\_loader = Data.DataLoader(dataset=train\_data,

batch\_size=64,

shuffle=True,

num\_workers=0 # 多线程容易报错

)

for step, (d\_x, d\_y) in enumerate(train\_loader):

if step > 0:

break

print('d\_x.shape:', d\_x.shape)

print('d\_y.shape:', d\_y.shape)

print('d\_x.dtype', d\_x.dtype)

print('d\_y.dtype', d\_y.dtype)

# d\_x.shape: torch.Size([64, 10])

# d\_y.shape: torch.Size([64])

# d\_x.dtype torch.float32

# d\_y.dtype torch.float32

''' # `load\_boston` has been removed from scikit-learn since version 1.2

报错 ，解决思路：上网下载后，本地加载 ；

换一个版本的sklearn ;

换一个数据集 。这里是换了一个数据集

D:\anaconda3\envs\portfolio\python.exe D:\pythoncode\learn\a\deep\_learning2.5.py

Traceback (most recent call last):

File "D:\pythoncode\learn\a\deep\_learning2.5.py", line 7, in <module>

from sklearn.datasets import load\_iris , load\_boston

File "D:\anaconda3\envs\portfolio\lib\site-packages\sklearn\datasets\\_\_init\_\_.py", line 157, in \_\_getattr\_\_

raise ImportError(msg)

ImportError:

`load\_boston` has been removed from scikit-learn since version 1.2.

The Boston housing prices dataset has an ethical problem: as

investigated in [1], the authors of this dataset engineered a

non-invertible variable "B" assuming that racial self-segregation had a

positive impact on house prices [2]. Furthermore the goal of the

research that led to the creation of this dataset was to study the

impact of air quality but it did not give adequate demonstration of the

validity of this assumption.

The scikit-learn maintainers therefore strongly discourage the use of

this dataset unless the purpose of the code is to study and educate

about ethical issues in data science and machine learning.

In this special case, you can fetch the dataset from the original

source::

import pandas as pd

import numpy as np

data\_url = "http://lib.stat.cmu.edu/datasets/boston"

raw\_df = pd.read\_csv(data\_url, sep="\s+", skiprows=22, header=None)

data = np.hstack([raw\_df.values[::2, :], raw\_df.values[1::2, :2]])

target = raw\_df.values[1::2, 2]

Alternative datasets include the California housing dataset and the

Ames housing dataset. You can load the datasets as follows::

from sklearn.datasets import fetch\_california\_housing

housing = fetch\_california\_housing()

for the California housing dataset and::

from sklearn.datasets import fetch\_openml

housing = fetch\_openml(name="house\_prices", as\_frame=True)

for the Ames housing dataset.

[1] M Carlisle.

"Racist data destruction?"

<https://medium.com/@docintangible/racist-data-destruction-113e3eff54a8>

[2] Harrison Jr, David, and Daniel L. Rubinfeld.

"Hedonic housing prices and the demand for clean air."

Journal of environmental economics and management 5.1 (1978): 81-102.

<https://www.researchgate.net/publication/4974606\_Hedonic\_housing\_prices\_and\_the\_demand\_for\_clean\_air>

'''

iris\_x, iris\_y = load\_iris(return\_X\_y=True)

print('iris\_x.dtype:', iris\_x.dtype)

print('iris\_y.dtype:', iris\_y.dtype)

# iris\_x.dtype: float64

# iris\_y.dtype: int32

print('iris\_x.shape:', iris\_x.shape)

print('iris\_y.shape:', iris\_y.shape)

# iris\_x.shape: (150, 4)

# iris\_y.shape: (150,)

train\_xt = torch.from\_numpy(iris\_x.astype(np.float32))

train\_yt = torch.from\_numpy(iris\_y.astype(np.int64))

print('train\_xt.dtype:', train\_xt.dtype)

print('train\_yt.dtype:', train\_yt.dtype)

# train\_xt.dtype: torch.float32

# train\_yt.dtype: torch.int64

train\_data = Data.TensorDataset(train\_xt, train\_yt)

train\_loader = Data.DataLoader(dataset=train\_data,

batch\_size=10, shuffle=True,

# num\_workers = 1, # 使用两个线程可能会报错

num\_workers=0,

)

for step, (b\_x, b\_y) in enumerate(train\_loader):

if step > 0:

break

print('b\_x.shape', b\_x.shape)

print('b\_y.shape', b\_y.shape)

# b\_x.shape torch.Size([10, 4])

# b\_y.shape torch.Size([10])

print('b\_x.dtype:', b\_x.dtype)

print('b\_y.dtype:', b\_y.dtype)

# b\_x.dtype: torch.float32

# b\_y.dtype: torch.int64

# File "D:\anaconda3\envs\portfolio\lib\site-packages\torch\utils\data\dataloader.py", line 1133, in \_try\_get\_data

# data = self.\_data\_queue.get(timeout=timeout)

# File "D:\anaconda3\envs\portfolio\lib\multiprocessing\queues.py", line 114, in get

# raise Empty

# \_queue.Empty

# import torch

# from torchvision.datasets import FashionMNIST

# import torchvision.transforms as transforms

# from torchvision.datasets import ImageFolder

#

# train\_data = FashionMNIST(root='./deep Learning/FashionMNIST',

# train=True,

# transform=transforms.ToTensor(),

# download=True)

#

# train\_loader = Data.DataLoader(dataset=train\_data,

# batch\_size=64,

# shuffle=True,

# num\_workers=0,

# )

#

# print(len(train\_loader))

# # 938

#

# test\_data = FashionMNIST(root='./deep Learning/FashionMNIST',

# train=False,

# download=False)

#

# test\_data\_x = test\_data.data.type(torch.FloatTensor) / 255.0

# test\_data\_x = torch.unsqueeze(test\_data\_x, dim=1)

# test\_data\_y = test\_data.targets

# print(test\_data\_x.shape)

# print(test\_data\_y.shape)

# # torch.Size([10000, 1, 28, 28])

# # torch.Size([10000])

#

#

# train\_data\_transforms = transforms.Compose([

# transforms.RandomResizedCrop(224),

# transforms.RandomHorizontalFlip(),

# transforms.ToTensor(),

# transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

# ])

# # transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

# # 是一个数据标准化（Normalization）操作，

# # 用于对图像数据进行标准化处理。其中的六个数字分别代表均值和标准差。

#

#

# train\_data\_dir = './deep Learning/'

# train\_data = ImageFolder(train\_data\_dir,

# transform=train\_data\_transforms,

# )

#

# train\_data\_loader = Data.DataLoader(

# train\_data,

# batch\_size=4,

# shuffle=True

# )

#

# print(train\_data.targets)

# # [0]

#

# for step, (b\_x, b\_y) in enumerate(train\_data\_loader):

# if step > 0:

# break

#

# print(b\_x.shape)

# print(b\_y.shape)

# print(b\_x.min(), b\_x.max())

# # torch.Size([1, 3, 224, 224])

# # torch.Size([1])

# # tensor(-1.4329) tensor(2.6400)

from torchtext import data

import torchtext

mytokenize = lambda x: x.split()

TEXT = torchtext.data.Field(sequential=True,

tokenize=mytokenize,

use\_vocab=True,

batch\_first=True,

fix\_length=200)

LABEL = data.Field(sequential=False, use\_vocab=False,

pad\_token=None,

unk\_token=None

)

text\_data\_fields = [('label', LABEL), ('text', TEXT)]

traindata, testdata = data.TabularDataset.splits(

path='./deep Learning/',

format='csv',

train='train.csv',

fields=text\_data\_fields,

test='test.csv',

skip\_header=True)

print(traindata, testdata)

# <torchtext.data.dataset.TabularDataset object at 0x000001739709DC60> <torchtext.data.dataset.TabularDataset object at 0x000001739709D6C0>

print(len(traindata), len(testdata))

# 1 1

TEXT.build\_vocab(traindata, max\_size=1000, vectors=None)

train\_iter = data.BucketIterator(traindata, batch\_size=4)

test\_iter = data.BucketIterator(testdata, batch\_size=4)

for step, batch in enumerate(train\_iter):

if step > 0:

break

print(batch.label)

print(batch.text.shape)

# D:\anaconda3\envs\portfolio\lib\site-packages\torchtext\data\\_\_init\_\_.py:4: UserWarning:

# /!\ IMPORTANT WARNING ABOUT TORCHTEXT STATUS /!\

# Torchtext is deprecated and the last released version will be 0.18 (this one). You can silence this warning by calling the following at the beginnign of your scripts: `import torchtext; torchtext.disable\_torchtext\_deprecation\_warning()`

# warnings.warn(torchtext.\_TORCHTEXT\_DEPRECATION\_MSG)

# Traceback (most recent call last):

# File "D:\pythoncode\learn\a\deep\_learning2.5.py", line 239, in <module>

# TEXT = data.Field(sequential=True,

# AttributeError: module 'torchtext.data' has no attribute 'Field'

# 官方的releases里面看到了，

# 官方在0.9.0版本中将Field 等函数放进了legacy中，

# 在最新版的0.12.0中移除了这个文件夹。

# pip 0.9.0没有找到，最终pip install torchtext==0.6.0 成功解决