Data structures

Dimensionality of data structures

0 Dimensions (Scalars)

1 Dimension (Vectors)

2 Dimensions (Matrices)

3 Dimensions (RGB image)

```
#例如一张2x2像素的RGB图片
image = [
    [[255, 0, 0], [0, 255, 0]],   # 第一行
    [[0, 0, 255], [255, 255, 255]] # 第二行
    ]
'''
image[0][0] 是第1行第1列的像素，颜色为红色 [255, 0, 0]
image[0][1] 是第1行第2列的像素，颜色为绿色 [0, 255, 0]
image[1][0] 是第2行第1列的像素，颜色为蓝色 [0, 0, 255]
image[1][1] 是第2行第2列的像素，颜色为白色 [255, 255, 255]
'''
```

4 Dimensions (RGB images in batches)

```
import numpy as np
# 初始化一个5张100x100像素的RGB图片的批量，所有像素初始值为0（黑色）
batch_size = 5
height = 100
width = 100
channels = 3
batch_images = np.zeros((batch_size, height, width, channels), dtype=np.uint8)
# 设置第1张图片的第2行第3列的像素颜色为RGB(210, 201, 202)
batch_images[0][1][2] = [210, 201, 202]
'''这种四维数组表示法在机器学习和深度学习中非常常见，尤其是在处理图像数据时。例如，在使用卷积神经网络（CNN）训练时，图像数据通常会以这样的四维数组批量传递给模型。'''
```

5 Dimensions (Videos in batches)

```
import numpy as np
# 初始化一个3个视频，每个视频10帧，每帧64x64像素的RGB视频批量，所有像素初始值为0（黑色）
batch_size = 3
frames = 10
height = 64
width = 64
channels = 3
batch_videos = np.zeros((batch_size, frames, height, width, channels),
dtype=np.uint8)
# 设置第1个视频的第2帧第3行第4列的像素颜色为RGB(210, 201, 202)
batch_videos[0][1][2][3] = [210, 201, 202]
'''这种五维数组表示法在视频处理、训练和预测任务中非常常见，尤其是在处理视频数据的深度学习模型中。视频的批量处理可以提高训练和推理的效率，尤其是在处理大规模视频数据集时。'''
```

Data operations

```
import torch
import numpy
x = torch.arange(12)
print('x:',x)
print('x.shape:',x.shape)
print(x.numel()) # returns the number of elements in the tensor(int)
x_reshape = x.reshape(3, 4)
print(x_reshape)
print(x_reshape.shape)
print(x.numel()) # returns the number of elements in the tensor(int)
all_zeros = torch.zeros((2,3,4)) # 2 dimension 3 row 4 col tensor with all zeros
print(all_zeros)
all_ones = torch.ones((2,3,4)) # 2 dimension 3 row 4 col tensor with all ones
print(all_ones)
a = torch.tensor([1,2,3],)
b = torch.tensor([4,5,6])
print(a+b) # [5, 7, 9]
print(torch.exp(a)) # [2.71828175, 7.38905621, 20.08553692]
c = torch.arange(12, dtype=torch.float32,).reshape(3,4)
d = torch.arange(12, 24, dtype=torch.float32,).reshape(3,4) # 左闭右开
print(c)
print(d)
print(torch.cat([c,d],dim=0)) # 维度0表示按行拼接，1表示按列拼接
print(torch.cat([c,d],dim=1))
demo = [[1,2,3],[4,5,6]] # List of lists
demo_numpy = numpy.array(demo)  # Convert list of lists to numpy array
demo_tensor = torch.tensor(demo) # Convert list of lists to tensor
print(demo_numpy)
print(demo_tensor)
e = torch.tensor([3.5])
print(e)
print(e.item()) # return the value of tensor as a Python scalar
print(type(e))
print(type(e.item())) # return the type of the scalar value
```

Data preprocessing

Create dataset

```
# 创建一个人工数据集，并存储在csv(字符分隔符文件)
import os
os.makedirs(os.path.join("..", "Deeplearning_Li Mu_Date"), exist_ok=True) # 创建文件夹
data_file = os.path.join("..", "Deeplearning_Li Mu_Date", "data.csv") # 文件路径
with open(data_file, "w") as f:
    f.write('NumRooms,Alley,Price\n')
    f.write('NA,Pave,127500\n')
    f.write('2,NA,106000\n')
    f.write('3,NA,178100\n')
    f.write('4,NA,140000\n')
'''
with open(data_file, "w") as f:是Python中的一种用于文件操作的语法结构，通常被称为上下文管理器(Context Manager)。
with 语句：引入上下文管理器，确保在退出代码块时，文件资源能够被正确释放。
open(data_file, "w")：打开或创建文件 data_file。
```

```
data_file是文件路径。
"w"是模式字符串，表示写入模式(write mode)。
"w"模式会覆盖文件中的现有内容。如果文件不存在，会创建一个新文件。
其他常见的模式包括 "r"(读取模式)、"a"(追加模式)、"b"(二进制模式)等。
as f：将文件对象(由open函数返回)赋给变量 f，用于在代码块中引用这个文件对象。
'''
```

Load dataset

```
import pandas as pd
import os
'''创造数据集'''
data_file = os.path.join('.','01_Data','01_house_tiny.csv')
data = pd.read_csv(data_file)
print(data)
'''读取一个CSV文件，并将其内容加载到一个Pandas DataFrame(data)中，然后打印出DataFrame的内容。'''
```

Preprocess dataset

```
import pandas as pd
import os
'''创造数据集+加载数据集'''
inputs, outputs = data.iloc[:,0:2], data.iloc[:,2] # # inputs取前两列，outputs取第三列
inputs = inputs.fillna(inputs.mean()) # 对 NaN 值用均值插值
print(inputs)
# 如有BUG，可尝试以下代码
# inputs['NumRooms'] = pd.to_numeric(inputs['NumRooms'], errors='coerce')
# numeric_columns = inputs.select_dtypes(include=[np.number]).columns
# inputs[numeric_columns] =
inputs[numeric_columns].fillna(inputs[numeric_columns].mean())
# print(inputs)
```

Convert dataset to tensor

```
import pandas as pd
import os
import torch
'''创造数据集+加载数据集+预处理数据集'''
inputs_tensor, outputs_tensor = torch.tensor(inputs.values),
torch.tensor(outputs.values)
print(inputs_tensor)
print(outputs_tensor)
```