

数据操作

1. 数组介绍

N维数组是机器学习和神经网络的主要数据结构

0维——标量

1维——向量

2维——矩阵

3维——RGB图片（宽 高 通道）

4维——一个RGB图片批量（批量大小 宽 高 通道）

5维——一个视频批量（批量大小 时间 宽 高 * 通道）

创建数组需要指定：形状，元素的数据类型，元素的值

访问元素索引（二维数组为例子）

#单个元素索引

一个元素--arr[1, 2]

#冒号：表示此行/列全部

一行--arr[1, :]

一列--arr[:, 1]

#**i:j**, 表示区间[i, j)对应索引，左闭右开，**i**:表示从*i*到该行/列末的索引

子区域--arr[1:3, 1:]

#**::i**, 表示跳跃索引，从零开始每隔*i*个元素

子区域--arr[::3, ::2]

2. torch基础张量运算

pytorch的包叫做torch

使用torch生成的叫做张量，张量表示一个数值组成的数组，可能有多个维度

```
## 导入pytorch对应的包torch
import torch
## 生成一个0到11的向量
x=torch.arange(12)
## 返回张量的形状
x.shape
## 返回张量元素数量
x.numel()
## 改变张量的形状将其变为3*4矩阵
X=x.reshape(3,4)
## 创建全0张量
torch.zeros((2,3,4))
## 创建全1张量
torch.ones((2,3,4))
## 指定特定元素值
torch.tensor([[1,2,3,4],[2,1,3,4],[4,3,2,1]]).shape
## 算数运算
x=torch.tensor([1.,2,4,8])
y=torch.tensor([2,2,2,2])
x+y,x-y,x*y,x/y
## 更多运算
torch.exp(x)
## 张量连接, dim=0是按行连接, dim=1是按照列连接
X=torch.arange(12,dtype=torch.float32).reshape((3,4))
Y=torch.tensor([[2.,1,4,3],[1,2,3,4],[4,3,2,1]])
torch.cat((X,Y),dim=0),torch.cat((X,Y),dim=1)
## 逻辑判断, 返回的是布尔型张量
X==Y
## 所有元素值求和
X.sum()
## 广播机制, a和b的形状不一样 , 将a和b均复制变为同样形状的矩阵再进行相加
a=torch.arange(3).reshape((3,1))
b=torch.arange(2).reshape((1,2))
a+b
```

```

## 索引访问
X[-1], X[1:3]
X[1, 2]=9
X[0:2, :]=12
## 内存地址更改
before=id(Y)
Y=Y+X ##将X+Y重新赋值给Y，此时Y的地址更改
id(Y)==before
## 原地修改值
Z=torch.zeros_like(Y) ##Y复制一个到Z
print('id(Z):', id(Z))
Z[:]=X+Y ##使用索引赋值不更改内存地址
print('id(Z):', id(Z))

before=id(X)
X+=Y ##自加运算也不更改内存地址
id(X)==before
## 转化为numpy数组
A=X.numpy()
B=torch.tensor(A)
type(A), type(B)

a=torch.tensor([3.5])
a, a.item(), float(a), int(a)

```

3.数据预处理

1. 得到原始数据文件，如果没有就创建

```

## 创建人工数据集
import os
os.makedirs(os.path.join('..', 'data'), exist_ok=True) #
创建目录
data_file=os.path.join('..', 'data', 'house_tiny.csv') #
创建csv文件
with open(data_file, 'w') as f: #写入文件

```

```
f.write('NumRooms,Alley,Price\n')
f.write('NA,Pave,127500\n')
f.write('2,NA,106000\n')
f.write('4,NA,178100\n')
f.write('NA,NA,140000\n')
```

2. 对原始数据文件进行读取和数据类型转换

```
import pandas as pd
data=pd.read_csv(data_file) #使用pandas中的read_csv函数来
#读取文件
inputs,outputs=data.iloc[:,0:2],data.iloc[:,2]
inputs['NumRooms'] = pd.to_numeric(inputs['NumRooms'])
#转为数值类型
```

3. 处理缺失值

```
#均值填充
inputs['NumRooms']=inputs['NumRooms'].fillna(inputs['Nu
mRooms'].mean())
#将含有缺失值的字符串列分为两列01数据进行输出
inputs=pd.get_dummies(inputs,dummy_na=True).astype(int)
```

4. 接收数据

```
import torch
x,y=torch.tensor(inputs.values,dtype=float),torch.tenso
r(outputs.values,dtype=float)
```