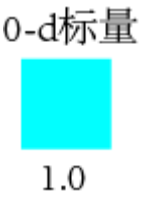
2 数据操作和数据预处理

1 N维数组样例

N维数组是机器学习和神经网络最主要的数据样式。

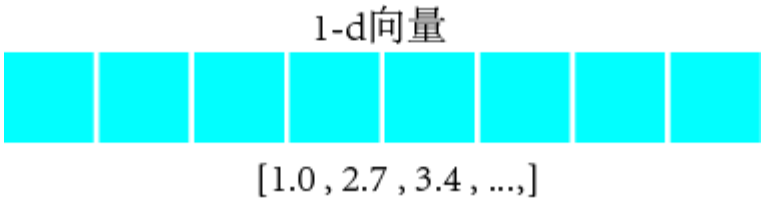
1.1 标量

标量是一个数据也可以说是一个元素，标量是0维的。



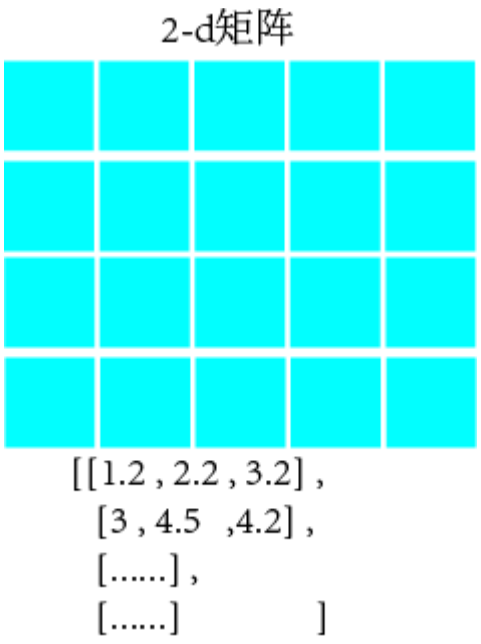
1.2 向量

向量一般是很多个数据的集合，也就是很多个标量的一个组合，向量是一维的。



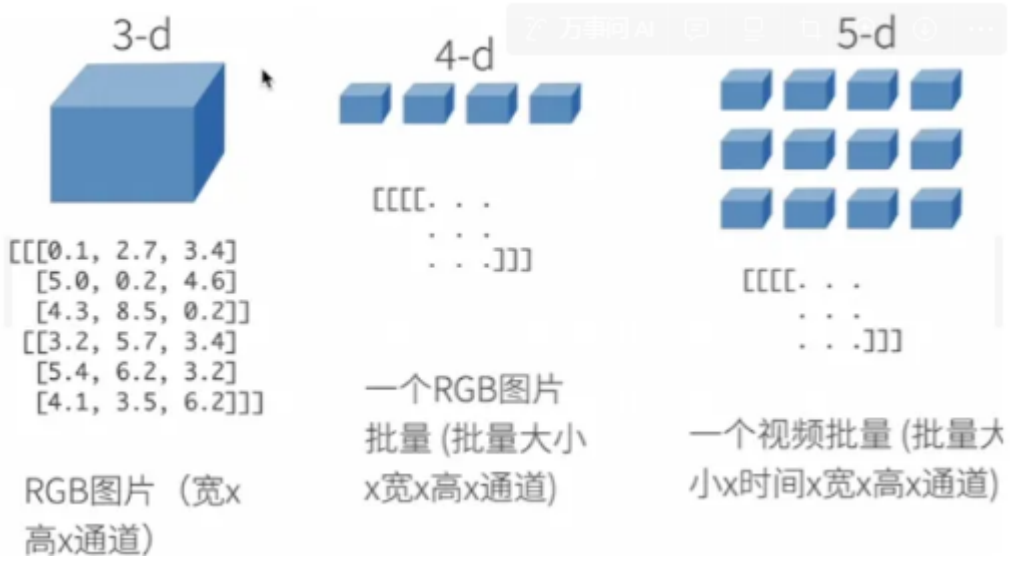
1.3 矩阵

矩阵是很多个向量的一个集合，矩阵是二维的。



1.4 其他

这里主要就是一些3到5维的一些数组样例

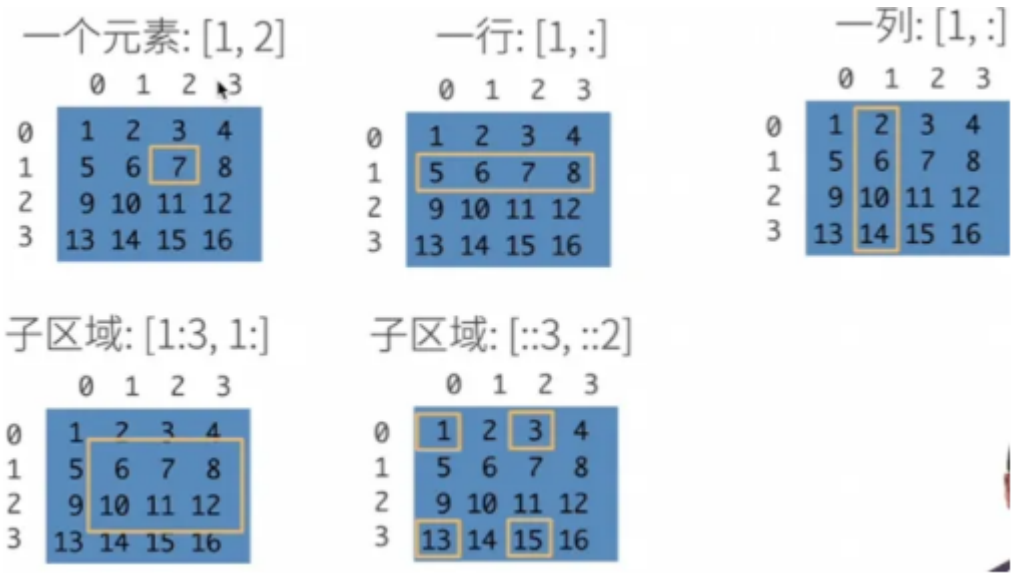


通常在进行学习的过程中最主要的就是 `4d` 的一个数组也就是一堆 `RGB` 的图片。

2 数组的创建

- 形状
- 元素的数据类型
- 每个元素的具体数据

3 访问元素




```
对于二维数组而言\\
一个元素的访问 &: [1,2]\\
一行元素的访问 &: [1:,]\\
一列元素的访问 &: [,:1]\\
子区域的访问 &: [1:3,1:]\\
子区域的访问&: [::3,::2]\\
```

4 数据操作

[2.1. 数据操作 — 动手学深度学习 2.0.0 documentation]

在进行操作之前需要先引入`pytorch`库函数

```
import torch
```

引入好了之后便可以进行相对应的一些操作了

```
x = torch.arange(12) #生成一个组单位向量

x.shape #访问向量的形状

x.numel() #元素的个数

x.reshape(3,4) #更改向量的类型 这里变成了三行四列了

torch.zeros((2,3,4)) #创建全零的张量

torch.ones((3,4,5)) #创建全一的张量

torch.tensor([[2,1,3,2],[4,5,3,5]]) #为张量的每一个元素进行赋值

x=torch.tensor([1,2,3,4])
y=torch.tensor([2,2,2,2])
```

```
x+y,x-y,x*y,x/y,x**y
#可以对元素进行数学的运算 当然是元素对元素的计算

x = torch.arrage(12,dtype=torch.float32).reshape((3,4))
y=torch.tensor([[2,3,4,5],[1,2,3,4],[2,4,5,6]])
torch.cat((x,y),dim=0)
torch.cat((x,y),dim = 1)
#将张量进行合并 0是按行合并 1是按列合并

x==y
#用逻辑运算符构建二维张量

x.sum()
#求和 只有一个元素的标量了

a = torch.arange(3).reshape((3,1))
b = torch.arange(2).reshape((1,2))
a+b
#形状不一样的相加怎么办呢，直接变成最高的一个张量就是直接就三行两列了
简单的说就是将a复制成一个三成二的矩阵
b复制成一个三成二的矩阵

x[-1] , x[1:3] , x[1,2]
#元素的访问

x[0:2,:] = 12
#多个元素的赋值 区域赋值

before = id(Y)
Y=Y+X
id(Y) == before
#相当于C的指针 id相当于取地址符 析构 难度比较高

A = X.numpy()
B=torch.tensor(A)
type(A) , type(B)
#将torch的转换成numpy的张量

a = torch.tensor([3.5])
a,a.item(),float(a),int(a)
#将大小为1的一个标量转换成python的量
```

5 数据的预处理

[2.2. 数据预处理 — 动手学深度学习 2.0.0 documentation]

列举完了`pytorch`的相关内容，现在我将数据预处理的东西也在下面进行处理以及举例

```
import os
import pandas as pd
!pip install pandas
data = pd.read_csv(data_file)
print(data)
```

```

os.makedirs(os.path.join('..' , 'data') , exist_ok = True)
data_file = os.path.join('..' , 'data' , 'house_tiny.csv')
with open(data_file , 'w') as f:
    f.write('NumRooms , Alley,price\n') #列名
    f.write('NA , Pave , 127500\n') #每行表示一个数据样本
    f.write('2 , NA , 106000\n')
    f.write('2 , NA , 178100\n')
    f.write('NA , NA , 140000\n')
#并从创建的csv文件中加载原始的数据集
inputs,outputs = data.iloc[: , 0:2],data.iloc[:2]
inputs = inputs.fillna(inputs.mean())
```

在使用的过程中会进行适当的数据补齐的处理方法，而这主要有两种方法:

1. 直接删掉当前行或者当前列
2. 对缺失的数据进行适当的插值处理

```
inputs = pd.get_dummies(inputs , dummy_na = True)
    print(inputs)
   #对于inputs中的类别值或离散值，我们将NAN视为一个类别
   import torch
   x , y = torch.tensor(inputs.values) , torch.tensor(outputs.values)
   x,y
```

至此现在inputs和outputs中的所有条目都是数值的类型，他们可以转换为张量的格式

一般在深度学习之中我们用的是32位浮点型