

深度学习

目标：介绍深度学习经典和最新的模型：Lenet，ResNet等

内容：

- 深度学习基础 — 线性神经网络，多层感知机
- 卷积神经网络 — LeNet, AlexNet, VGG, Inception, ResNet
- 循环神经网络 — RNN, GRU, LSTM, seq2seq
- 注意力机制 — Attention, Transformer
- 优化算法 — SGD, Momentum, Adam
- 高性能计算 — 并行，多GPU，分布式
- 计算机视觉 — 目标检测，语义分割
- 自然语言处理 — 词嵌入，BERT

资源

资源

- 课程主页：<https://courses.d2l.ai/zh-v2>
- 教材：<https://zh-v2.d2l.ai/>
- 课程论坛讨论：<https://discuss.d2l.ai/c/16>
- Pytorch论坛：<https://discuss.pytorch.org/>

深度学习的介绍

有效≠可解释性

领域专家：甲方；

数据科学家：乙方；

数据操作

机器学习和神经网络的主要数据结构是N维数组

实现

[d2l-zh-pytorch-slides/ndarray.ipynb at main · d2l-ai/d2l-zh-pytorch-slides \(github.com\)](#)

张量表示一个数值组成的数组，这个数组可能有多个维度

```
In [2]: x = torch.arange(12)
x
Out[2]: tensor([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

#####

reshape(): 改变x的形状，不改变值

初始化全0: torch.zeros()

通过提供包含数值的 Python 列表（或嵌套列表）来为所需张量中的每个元素赋予确定值

```
In [3]: tensor([[[2, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2, 1]]])  
Out[3]: tensor([[[2, 1, 4, 3],  
                [1, 2, 3, 4],  
                [4, 3, 2, 1]]])
```

$x**y$ 表示求幂运算

我们也可以把多个张量连结在一起

```
In [12]: X = torch.arange(12, dtype=torch.float32).reshape((3, 4))  
         Y = torch.tensor([2.0, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2, 1])  
         torch.cat((X, Y), dim=0), torch.cat((X, Y), dim=1)  
Out[12]: (tensor([[ 0.,  1.,  2.,  3.],  
                  [ 4.,  5.,  6.,  7.],  
                  [ 8.,  9., 10., 11.],  
                  [ 2.,  1.,  4.,  3.],  
                  [ 1.,  2.,  3.,  4.],  
                  [ 4.,  3.,  2.,  1.]]),  
          tensor([[ 0.,  1.,  2.,  3.,  2.,  1.,  4.,  3.],  
                  [ 4.,  5.,  6.,  7.,  1.,  2.,  3.,  4.]])
```

dim=0表示第0维合并（行），=1表示列

$x==y$ 对每个元素进行判断

即使形状不同，我们仍然可以通过调用 **广播机制**（broadcasting mechanism）来执行按元素操作

```
In [15]: a = torch.arange(3).reshape((3, 1))  
         b = torch.arange(2).reshape((1, 2))  
         a, b
```

```
Out[15]: (tensor([[0],  
                  [1],  
                  [2]]),  
          tensor([[0, 1]]))
```

```
In [16]: a + b
```

```
Out[16]: tensor([[0, 1],  
                 [1, 2],  
                 [2, 3]])
```

广播机制，a 1->2, b 1->3 形成一个3*2矩阵（注意，容易出错）

为多个元素赋值相同的值，我们只需要索引所有元素，然后为它们赋值。

```
In [19]: x[0:2, :] = 12
x
Out[19]: tensor([[12., 12., 12., 12.],
                 [12., 12., 12., 12.],
                 [ 8.,  9., 10., 11.]])
```

x = 0; x < 2; 赋值

数据预处理

[d2l-zh-pytorch-slides/pandas.ipynb at main · d2l-ai/d2l-zh-pytorch-slides \(github.com\)](https://github.com/d2l-ai/d2l-zh-pytorch-slides/blob/main/pandas.ipynb)

关于浅复制：

```
In [6]: a = torch.arange(12)
        b = a.reshape((3,4))
        b[:, :] = 2
        a
Out[6]: tensor([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

线性代数

[d2l-zh-pytorch-slides/linear_algebra.ipynb at main · d2l-ai/d2l-zh-pytorch-slides \(github.com\)](https://github.com/d2l-ai/d2l-zh-pytorch-slides/blob/main/linear_algebra.ipynb)

指定张量沿哪一个轴来通过求和降低维度

```
In [16]: A_sum_axis0 = A.sum(axis=0)
        A_sum_axis0, A_sum_axis0.shape
```

```
Out[16]: (tensor([40., 45., 50., 55.]), torch.Size([4]))
```

```
In [17]: A_sum_axis1 = A.sum(axis=1)
        A_sum_axis1, A_sum_axis1.shape
```

```
Out[17]: (tensor([ 6., 22., 38., 54., 70.]), torch.Size([5]))
```

```
In [18]: A.sum(axis=[0, 1])
```

```
Out[18]: tensor(190.)
```

计算总和或均值时保持轴数不变

```
In [21]: sum_A = A.sum(axis=1, keepdims=True)
          sum_A
```

```
Out[21]: tensor([[ 6.],
                  [22.],
                  [38.],
                  [54.],
                  [70.]])
```

通过广播将 A 除以 sum_A

```
In [22]: A / sum_A
```

```
Out[22]: tensor([[0.0000, 0.1667, 0.3333, 0.5000],
                  [0.1818, 0.2273, 0.2727, 0.3182],
                  [0.2105, 0.2368, 0.2632, 0.2895],
                  [0.2222, 0.2407, 0.2593, 0.2778],
                  [0.2286, 0.2429, 0.2571, 0.2714]])
```


我是这样理解的：按照哪个轴求和就会丢掉该维度， $axis=0$ 就丢掉第一个维度5，保留4。那么只有按列求和 $axis=1$ 等

红字大佬
等于是1到5行相加
式是 数学求和时， i 和 j ，表示相同行，相同列

轴数是矩阵中向量的维数
用-1，-2更简单直接

想象一下是把行折叠起来，这不就是按列求和
 sum 的 $axis$ 为0（行），不就是把各行加起来吗
以哪个 $axis$ 求 就把对应的 $axis$ 看成一个整体（一个行）

0



sum

shape: [5, 4]

axis: 0, 1

$$ax' \leq 0, \text{ Sum: } [4]$$
$$ax^i \leq 1, \text{ Sum}$$


axis=0, 不是按照行求和 这里的

keepdims = 1

Shape [5, 4]

axis: 1 [5,]

axis: 0 [4]

Shape: [2, 5, 4]

axis: 1 [2, 1, 4]


axis: [1, 2] [2, 1, 1]

Shape [2, 5, 4]

axis: 1 [2, 4]

axis: 2 [2, 5]

axis: [1, 2] [4]



关于向量求导

这里最好加一个括号比较好
一个值, 向量是一串值, 二维向量就是矩阵



求偏导吧
y的表达式加括号好理解

有点偏导数的意思了
为什么向量在分母, 就转置?
这里不需要括号括起来吗?

其他不想管

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \frac{\partial y}{\partial \mathbf{x}} = \left[\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \dots, \frac{\partial y}{\partial x_n} \right]$$

$\frac{\partial}{\partial \mathbf{x}} x_1^2 + 2x_2^2 = [2x_1, 4x_2]$ 方向 (2, 4) 跟等高线正交

$$\partial \mathbf{y} / \partial \mathbf{x}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial \mathbf{x}} \\ \frac{\partial y_2}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial y_m}{\partial \mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1}, \frac{\partial y_1}{\partial x_2}, \cdots, \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1}, \frac{\partial y_2}{\partial x_2}, \cdots, \frac{\partial y_2}{\partial x_n} \\ \vdots \\ \frac{\partial y_m}{\partial x_1}, \frac{\partial y_m}{\partial x_2}, \cdots, \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$