



# ExerciseThinking

## 卷二 动手学深度学习 Pytorch 习题解

作者：latalealice

日期：2025/04/14

# 目 录

序言 .....	1
第一章 引言 .....	2
第二章 预备知识 .....	3
2.1 数据操作 .....	3
2.2 数据预处理 .....	3
2.3 线性代数 .....	4

## 序言

此 PDF 为习题解答,需求的库默认 import

# 第一章 引言

1. 你当前正在编写的代码的哪些部分可以“学习”,即通过学习和自动确定代码中所做的设计选择来改进?你的代码是否包含启发式设计选择?
2. 你遇到的哪些问题有许多解决它们的样本,但没有具体的自动化方法?这些可能是使用深度学习的主要候选者.
3. 如果把人工智能的发展看作一场新的工业革命,那么算法和数据之间的关系是什么?它类似于蒸汽机和煤吗?根本区别是什么?
4. 你还可以在哪里应用端到端的训练方法,比如 图 1.1.2 、物理、工程和计量经济学?

## 第二章 预备知识

### 2.1 数据操作

1. 运行代码,将条件语句  $X == Y$  更改为  $X < Y$  或  $X > Y$ ,看看可以得到什么样的张量

代码

```
1 X = torch.arange(12,  
dtype=torch.float32).reshape((3,4))  
2 Y = torch.tensor([[2.0, 1, 4, 3], [1, 2,  
3, 4], [4, 3, 2, 1]])  
3 X > Y, X == Y, X < Y
```

结果

```
1 tensor([[[False, False, False, False],  
2 [ True,  True,  True,  True],  
3 [ True,  True,  True,  True]]],  
4 tensor([[[False, True, False, True],  
5 [False, False, False, False],  
6 [False, False, False, False]]],  
7 tensor([[[ True, False, True, False],  
8 [False, False, False, False],  
9 [False, False, False, False]])
```

2. 用其他形状(例如三维张量)替换广播机制中按元素操作的两个张量,结果如何

代码

```
1 a =  
torch.arange(25).reshape((-1,5,1))  
2 b = torch.arange(10).reshape((-1,1,2))  
3 a+b  
4 # 广播矩阵,复制a的列,b的行
```

结果

```
1 tensor([[[[ 0,  1],  
2 [ 1,  2],  
3 [ 2,  3],  
4 [ 3,  4],  
5 [ 4,  5]],  
6 [[ 7,  8],  
7 [ 8,  9],  
8 [ 9, 10],  
9 [10, 11],  
10 [11, 12]],  
11 [[14, 15],  
12 [15, 16],  
13 [16, 17],  
14 [17, 18],  
15 [18, 19]],  
16 [[21, 22],  
17 [22, 23],  
18 [23, 24],  
19 [24, 25],  
20 [25, 26]],  
21 [[28, 29],  
22 [29, 30],  
23 [30, 31],  
24 [31, 32],  
25 [32, 33]]]])
```

### 2.2 数据预处理

创建包含更多行和列的原始数据集

- 删除缺失值最多的列

- 将预处理后的数据集转换为张量格式

代码

```

1 os.makedirs(os.path.join '..',
  'data'), exist_ok=True)
2 data_file = os.path.join '..', 'data',
  'house_tiny.csv')
3 with open(data_file, 'w') as f:
4     f.write('NumRooms,Alley,Price\n') # 列
    名
5     f.write('NA,Pave,127500\n') # 每行表示
    一个数据样本
6     f.write('2,NA,106000\n')
7     f.write('4,NA,178100\n')
8     f.write('NA,NA,140000\n')
9     f.write('3,Pave,122000\n')
10    # 1. 删除缺失值最多的列
11    # 计算每列的缺失值数量
12    missing_values = data.isnull().sum()
13    # 找出缺失值最多的列名
14    column_to_drop =
    missing_values.idxmax()
15    # 删除该列
16    data_preprocessed =
    data.drop(column_to_drop, axis=1)
17    print(data_preprocessed)
18    # 2. 将预处理后的数据集转换为张量格式
19    # 首先处理剩余的缺失值 (用均值填充数值列)
20    data_preprocessed['NumRooms'] =
    data_preprocessed['NumRooms'].
    fillna(data_preprocessed['NumRooms'].mean())
21    # 转换为张量
22    tensor_data =
    torch.tensor(data_preprocessed.values,
    dtype=torch.float32)
23    print(tensor_data)
  
```

结果

	NumRooms	Price
1	0	NaN
2	1	2.0
3	2	4.0
4	3	NaN
5	4	3.0
6	NaN	127500
7	2.0	106000
8	4.0	178100
9	NaN	140000
10	3.0	122000
11	tensor([[3.0000e+00, 1.2750e+05],	
12	[2.0000e+00, 1.0600e+05],	
13	[4.0000e+00, 1.7810e+05],	
14	[3.0000e+00, 1.4000e+05],	
15	[3.0000e+00, 1.2200e+05]])	

## 2.3 线性代数

1. 证明一个矩阵  $A$  的转置的转置是  $A$

代码

```

1 A = torch.arange(20).reshape(5, -1)
2 A.T.T == A
  
```

结果

1	tensor([[True, True, True, True],
2	[True, True, True, True],
3	[True, True, True, True],
4	[True, True, True, True],
5	[True, True, True, True]])

2. 给出两个矩阵  $A$  和  $B$ , 证明“它们转置的和”等于“它们和的转置”

代码

```
1 A = torch.randn(3,4)
2 B = torch.randn(3,4)
3 (A+B).T == A.T+B.T
```

py

结果

```
1 tensor([[True, True, True],
2         [True, True, True],
3         [True, True, True],
4         [True, True, True]])
```

py

3. 给定任意方阵  $A$ ,  $A + A^T$  总是对称的吗

代码

```
1 A=torch.randn(4,4)
2 A+A.T == (A.T+A).T
```

py

结果

```
1 tensor([[True, True, True, True],
2         [True, True, True, True],
3         [True, True, True, True],
4         [True, True, True, True]])
```

py

4. 定义形状(2,3,4)的张量  $X$ ,  $\text{len}(X)$  的输出结果是什么

代码

```
1 X = torch.arange(24).reshape(2,3,4)
2 len(X)
```

py

结果

```
1 2
```

py

5. 对于任意形状的张量  $X$ ,  $\text{len}(X)$  是否总是对应于  $X$  特定轴的长度?这个轴是什么

代码

```
1 # 1维张量
2 X = torch.tensor([1, 2, 3])
3 len(X)
4 # 2维张量
5 X = torch.tensor([[1, 2], [3, 4], [5, 6]])
6 len(X)
7 # 3维张量
8 X = torch.rand(2, 3, 4)
9 len(X)
```

py

结果

```
1 # len(X)总是返回张量在第一个轴axis=0的
   长度
2 3
3 3
4 2
```

py

6. 运行  $A/A.\text{sum}(\text{axis}=1)$ , 看看会发生什么。请分析一下原因

代码

```
1 A=torch.arange(25).reshape(5,-1)
2 A
3 A/A.sum(axis=1)
4 tensor([[ 0,  1,  2,  3,  4],
5         [ 5,  6,  7,  8,  9],
6         [10, 11, 12, 13, 14],
7         [15, 16, 17, 18, 19],
8         [20, 21, 22, 23, 24]])
```

py

结果

```
1 tensor([
2         [0.0000, 0.0286, 0.0333, 0.0353,
3           0.0364],
4         [0.5000, 0.1714, 0.1167, 0.0941,
5           0.0818],
6         [1.0000, 0.3143, 0.2000, 0.1529,
7           0.1273],
8         [1.5000, 0.4571, 0.2833, 0.2118,
9           0.1727],
10        [2.0000, 0.6000, 0.3667, 0.2706,
11          0.2182]])
```

py

7. 考虑一个具有形状(2,3,4)的张量,在轴 0、1、2 上的求和输出是什么形状

代码

```
1 A = torch.arange(24).reshape(2,3,4)
2 A
3 A.sum(axis=0)
4 A.sum(axis=1)
5 A.sum(axis=2)
```

结果

```
1 tensor([[12, 14, 16, 18],
2         [20, 22, 24, 26],
3         [28, 30, 32, 34]])
4 tensor([[12, 15, 18, 21],
5         [48, 51, 54, 57]])
6 tensor([[ 6, 22, 38],
7         [54, 70, 86]])
```

8. 为 linalg.norm 函数提供 3 个或更多轴的张量,并观察其输出.对于任意形状的张量这个函数计算得到什么

代码

```
1 # 创建一个3x4x5的张量
2 tensor_3d = torch.randn(3, 4, 5)
3 norm_default = np.linalg.norm(tensor_3d)
4 norm_default
5 norm_axis0 = np.linalg.norm(tensor_3d,
6                               axis=0)
7 norm_axis01 = np.linalg.norm(tensor_3d,
8                               axis=(0, 1))
9 norm_axis01
```

结果

```
1 7.6230054
2 [[1.5946476  1.4472774  2.368117
3    2.1636908  2.045201 ],[1.8514549
4    1.5507743  2.2650828  0.7746272
5    1.3306752 ],[1.4223679  0.55377847
6    2.2253275  1.3022163  1.2005248 ],
7    [1.6133649  2.9035609  1.4079779
8    1.3459872  0.64774853]]
9 [4.924984 , 2.7686076, 3.7564898,
10  3.958195 , 2.5980809]
```