Guidebook to Exercise 2 (Tensors)

Prof. Dr. Ivan Habernal
Yassine Thlija

2025-10-22

1 Tensors: The Fundamental PyTorch Data Structure

A **tensor** is a generalization of vectors and matrices to an arbitrary number of dimensions.

- A **0-D tensor** is a scalar (`torch.tensor(5)`)
- A **1-D tensor** is a vector (`torch.tensor([1,2,3])`)
- A **2-D tensor** is a matrix
- A **3-D tensor** or higher is used for structured data (e.g., images, batches)

Tensors hold numerical data and are the foundation of all computations in PyTorch, from deep learning to numerical simulation.

2 Creating Tensors

Use `torch.tensor()` to create tensors directly from Python lists:

```
tensor = torch.tensor([1, 2, 3, 4, 5, 6])
```

Shape and Size

Each tensor has a **shape**, which defines its dimensionality:

```
tensor.shape        # torch.Size([6])
```

You can reshape tensors using:

`torch.reshape(tensor, (rows, cols))`

Example transformations:

- (2, 3) → 2 rows, 3 columns
- (3, 2) → 3 rows, 2 columns
- (3, 2, 1) → a 3-D tensor with 3 "blocks"

3 Tensor Initialization

PyTorch provides convenience functions for creating constant or random tensors:

Function	Description	Example
`torch.full(size, value)`	Tensor filled with a constant	`torch.full([2,3], 9)`
`torch.ones(size)`	Tensor of ones	`torch.ones([3,3])`
`torch.zeros(size)`	Tensor of zeros	`torch.zeros([3,3])`
`torch.arange(start, end)`	Sequence of numbers	`torch.arange(0,5)`

4 Stacking and Concatenation

`torch.stack()` joins multiple tensors along a new dimension.

```
x = torch.tensor([1, 4])
y = torch.tensor([2, 5])
z = torch.tensor([3, 6])
torch.stack([x, y, z])        # shape: (3, 2)
torch.stack([x, y, z], dim=1) # shape: (2, 3)
```

- `dim=0` adds a new row dimension
- `dim=1` stacks along columns

5 Indexing and Slicing

You can access elements or slices of tensors using Python indexing syntax. Examples:

Operation	Description	Output
`tensor[2:-2]`	Elements between indices 2 and -2	`[3, 4]`
`tensor[::2]`	Every other element	e.g., `[1, 3, 5]`
`tensor[torch.tensor(0)]`	First row	`[1, 2, 3]`
`tensor[:, torch.tensor(0)]`	First column	`[1, 4, 7]`

Equivalent forms:

```
tensor[0]
tensor[0, :]
tensor[0, ...]
```

6 Broadcasting

Broadcasting allows operations between tensors of different shapes by expanding one to match the other without copying data.

Expression	Description	Output shape
`a + b` (both scalars)	Simple addition	scalar
`a=[1,2], b=4`	b is broadcast to match a	`[5,6]`
`a=[[1,2,3],[4,5,6]], b=[10,20,30]`	b broadcast across rows	(2,3)
`a=[[1,2,3],[4,5,6]], b=[[1],[2]]`	b broadcast across columns	(2,3)

Broadcasting rules:

1. Start comparing shapes from the rightmost dimension
2. Dimensions must be equal, or one must be 1
3. Missing dimensions are treated as 1

7 Arithmetic and Matrix Operations

PyTorch supports all standard arithmetic operators:

Operation	Description
+,-,*,/,%,//	Element-wise operations
@ or `torch.matmul()`	Matrix multiplication

Example:

```
a = torch.tensor([[1, 2]])
b = torch.tensor([[3], [4]])
a @ b   # matrix multiplication → tensor([[11]])
```

8 Reduction Operations

Reduction operations collapse tensor dimensions by applying an aggregation function:

Function	Description	Example
`torch.max(t)`	Maximum element overall	tensor(6)
`torch.max(t, dim=0)`	Max along the first axis	values + indices
`torch.mean(t, dim=1)`	Mean along the second axis	[0.33, 4.00]
`torch.sum(x, dim=1, keepdim=True)`	Keeps reduced dimension	[[3],[3]]

9 Norms and Vector Lengths

The **L2 norm** (Euclidean length) of a vector measures its distance from the origin:

```
torch.linalg.vector_norm(x, ord=2, dim=1)
```

- With `dim`: computes per-row norm
- Without `dim`: reduces the entire tensor

10 Type Inference and Dtypes

Tensors can have different data types (`dtype`), such as:

- `torch.float32`, `torch.float64` (floating point)
- `torch.int32`, `torch.int64` (integers)

Be cautious: if you create a tensor from integers, operations like `torch.mean()` may produce integer outputs unless you specify `dtype=torch.float`.

Example:

```
torch.mean(torch.tensor([1, 0, 0], dtype=torch.float))
```