Deep Learning
Assignment 2

PyTorch Tensor Functions

P G Raveesha Dulmi
D/DBA/21/0017
CS4193

Faculty of Computing
Department of Computational Mathematics
Data Science and Business Analytics
General Sir John Kotelawala Defence University

Friday, 8 March 2024

PyTorch

PyTorch is a machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella.

Tensors

In deep learning higher-dimensional arrays are called tensors. A tensor is simply an n-dimensional array of numbers.

For example,

- vector is a one-dimensional tensor
- matrix is a two-dimensional tensor
- An image is a three-dimensional tensor (width, height, and depth)

PyTorch Tensor Functions

1. **Tensor Function**

- Creates a new tensor with the specified data.
- It takes in a variety of data types and builds a new tensor from them, including scalar values, NumPy arrays, Python lists, tuples, and other tensors.

```
import torch

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

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

2. **Trance Function**

- Computes the sum of the diagonal elements of a matrix of 2D tensor or matrix.
- It offers a practical means of calculating a matrix's trace, which is the total of the elements along its principal diagonal.

```
matrix = torch.tensor([[1, 2, 3],
                       [4, 5, 6],
                       [7, 8, 9]])
trace = torch.trace(matrix)
print(trace)
```

```
tensor(15)
```

3. Mean Function

- used to find the mean (average) value of tensor elements along given dimensions, or over all elements in the absence of a dimension.

```
tensor = torch.tensor([[1, 2, 3],
                       [4, 5, 6]], dtype=torch.float32)
mean = torch.mean(tensor)
print(mean)
```

```
tensor(3.5000)
```

4. Eye Function

- Creates a 2-D tensor with ones on the diagonal and zeros elsewhere.
- It creates an identity matrix with size n*n, where the user specifies n.
- The default data type of tensors is 32 bit float data type.

```
eye_tensor = torch.eye(3)
print(eye_tensor)
```

```
tensor([[1., 0., 0.],
        [0., 1., 0.],
        [0., 0., 1.]])
```

5. Full Function

- Creates a tensor filled with a specified value.

```
full_tensor = torch.full((2, 3), 7)
print(full_tensor)
```

```
tensor([[7, 7, 7],
        [7, 7, 7]])
```

- No of rows = 2
- No of columns = 3
- Specified value = 7

6. Log Function

- Computes the natural logarithm of tensor elements.

```
tensor = torch.tensor([1, 2, 3], dtype=torch.float32)
log_tensor = torch.log(tensor)
print(log_tensor)
```

```
tensor([0.0000, 0.6931, 1.0986])
```

- dtype - type of data stored in the tensor. In here the type is 32-bit floating-point data type.

7. Exp Function

- Computes the exponential of tensor elements.

```
tensor = torch.tensor([1, 2, 3], dtype=torch.float32)
exp_tensor = torch.exp(tensor)
print(exp_tensor)
```

```
tensor([ 2.7183,  7.3891, 20.0855])
```

8. Chunk Function

- Splits a tensor into a specified number of chunks along a specified dimension.

```
tensor = torch.tensor([[1, 2, 3],
                       [4, 5, 6],
                       [7, 8, 9]])
chunks = torch.chunk(tensor, chunks=2, dim=0)
print(chunks)
```

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

- Split the tensor into two chunks along dimension 0

9. Flatten Function

- Flattens a contiguous range of dimensions of a tensor.

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

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

10. Clone Function

- Creates a deep copy of the input tensor.
- It creates a new tensor with the same data and shape as the original tensor but separate memory allocation.

```
[15] tensor = torch.tensor([1, 2, 3])
     cloned_tensor = torch.clone(tensor)
     print(cloned_tensor)
```

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

11. Max Function

- Computes the maximum value of all elements in the input tensor.

```
tensor = torch.tensor([1, 5, 3, 7, 2])
max_value = torch.max(tensor)
print(max_value)
```

```
tensor(7)
```

12. Sin Function

- Computes the sin of the elements of the input tensor.

```
tensor = torch.tensor([0, 1, 2, 3, 4])
sin_tensor = torch.sin(tensor)
print(sin_tensor)
```

```
tensor([ 0.0000,  0.8415,  0.9093,  0.1411, -0.7568])
```

13. Round Function

- Rounds tensor elements to the nearest integer.

```
tensor = torch.tensor([1.3, 2.6, 3.9])
rounded_tensor = torch.round(tensor)
print(rounded_tensor)
```

```
tensor([1., 3., 4.])
```

14. Zeros Function

- Creates a tensor filled with zeros.

```
zeros_tensor = torch.zeros(2, 3)
print(zeros_tensor)
```

```
tensor([[0., 0., 0.],
        [0., 0., 0.]])
```

15. **Ones Function**

- Creates a tensor filled with ones.

```
ones_tensor = torch.ones(2, 3)
print(ones_tensor)
```

```
tensor([[1., 1., 1.],
        [1., 1., 1.]])
```

16. is_tensor Function

- Checks if the input is a PyTorch tensor.

```
tensor = torch.tensor([1, 2, 3])
print(torch.is_tensor(tensor))
```

```
True
```

17. Min Function

- Computes the minimum value of all elements in the input tensor.

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

min_value = torch.min(tensor)
print( min_value.item())
```

```
1
```

18. Hstack Function

- Stacks tensors horizontally.

```
tensor1 = torch.tensor([[1, 2],
                        [3, 4]])
tensor2 = torch.tensor([[5, 6],
                        [7, 8]])
hstacked_tensor = torch.hstack((tensor1, tensor2))
print(hstacked_tensor)
```

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

19. Swapaxes Function

- Swaps axes of a tensor.

```
tensor = torch.tensor([[1, 2, 3],
                       [4, 5, 6]])
swapped_tensor = torch.swapaxes(tensor, 0, 1)
print(swapped_tensor)
```

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

20. Reshape Function

- Reshapes a tensor to the specified shape.

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

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

21. **Linspace Function**

- Creates a one-dimensional tensor of equally spaced points.

```
tensor = torch.linspace(0, 10, steps=5)
print(tensor)
```

```
tensor([ 0.0000,  2.5000,  5.0000,  7.5000, 10.0000])
```

22. Randint Function

- Returns a tensor filled with random integers.

```
random_int_tensor = torch.randint(0, 10, (3, 3))
print(random_int_tensor)
```

```
tensor([[4, 6, 8],
        [0, 0, 7],
        [9, 5, 9]])
```

23. Matmul Function

- Performs matrix multiplication.

```
tensor1 = torch.tensor([[1, 2],
                        [3, 4]])
tensor2 = torch.tensor([[5, 6],
                        [7, 8]])
matmul_result = torch.matmul(tensor1, tensor2)
print(matmul_result)
```

```
tensor([[19, 22],
        [43, 50]])
```

24. Rand Function

- Returns a tensor filled with random numbers from a uniform distribution over [0, 1).

```
random_tensor = torch.rand(2, 3)
print(random_tensor)
```

```
tensor([[0.5004, 0.0602, 0.8368],
        [0.8054, 0.7880, 0.9990]])
```

25. Unbind Function

- Removes a tensor dimension.

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

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

References

https://dahalegopal27.medium.com/five-must-know-pytorch-tensor-functions-1a95426d6359

https://pytorch.org/docs/stable/torch.html

https://machinelearningmastery.com/manipulating-tensors-in-pytorch/

https://medium.com/analytics-vidhya/pytorch-for-deep-learning-part-1-1324a45b0af3