Abdullah Al Asif

2022521460130

Assignment-1

Screenshot of the calculation and explanation.

```
PS G:\5th Semester\Neural Modeling\Assignment-1> python -u "g:\5th Semester\Neural Modeling\Assignment-1\pytorch.py"
First row: tensor([1., 1., 1., 1.])
First column: tensor([1., 1., 1., 1.])
Last column: tensor([1., 1., 1., 1.])
tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.]])
Matrix multiplication results:
y1:
tensor([[3., 3., 3., 3.],
        [3., 3., 3., 3.],
        [3., 3., 3., 3.],
        [3., 3., 3., 3.]])
y2:
tensor([[3., 3., 3., 3.],
        [3., 3., 3., 3.],
        [3., 3., 3., 3.],
        [3., 3., 3., 3.]])
y3 (with out parameter):
tensor([[3., 3., 3., 3.],
        [3., 3., 3., 3.],
        [3., 3., 3., 3.],
        [3., 3., 3., 3.]])

Element-wise multiplication results:
z1:
tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.]])
z2:
tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.]])
z3 (with out parameter):
tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.]])
PS G:\5th Semester\Neural Modeling\Assignment-1>
```

1. **Tensor Initialization**:

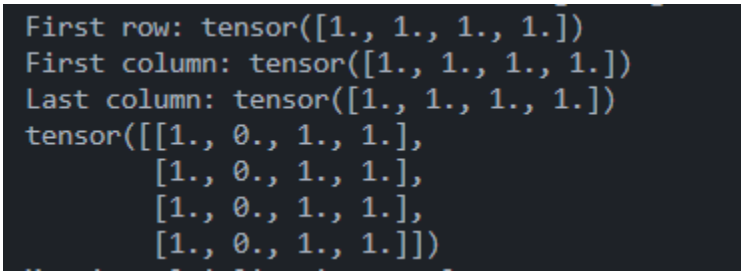
`tensor = torch.ones(4, 4)` creates a 4x4 matrix (tensor) filled with ones.

2. **Accessing Specific Parts of the Tensor**:

- `tensor[0]` extracts the first row: `[1., 0., 1., 1.]`.
- `tensor[:, 0]` extracts the first column: `[1., 1., 1., 1.]`.
- `tensor[..., -1]` extracts the last column: `[1., 1., 1., 1.]`.

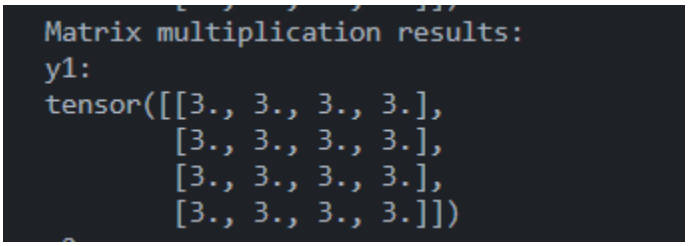
3. **Setting the Second Column to Zero**:

`tensor[:, 1] = 0` modifies the tensor by setting the second column to zero. The tensor now looks like:



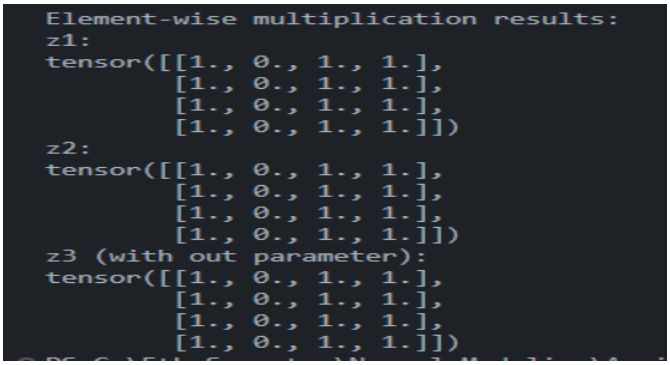
Matrix Multiplication:

- `y1 = tensor @ tensor.T` and `y2 = tensor.matmul(tensor.T)` both compute the matrix multiplication between `tensor` and its transpose `tensor.T`.
- `torch.matmul(tensor, tensor.T, out=y3)` performs the same matrix multiplication but stores the result in `y3`.
- All three (y1, y2, and y3) will have the same result since they perform the same operation. The result will be a 4x4 matrix:



Element-wise Multiplication:

- `z1 = tensor * tensor` and `z2 = tensor.mul(tensor)` both perform element-wise multiplication of the tensor with itself.
- `torch.mul(tensor, tensor, out=z3)` performs the same operation but stores the result in `z3`.
- The result of the element-wise multiplication will be



Differences between Matrix Multiplication and Element-wise Multiplication:

- **Matrix Multiplication**: Combines rows of the first matrix with columns of the second matrix following matrix multiplication rules. In this case, multiplying the tensor by its transpose results in summing products of the corresponding row and column elements.
- **Element-wise Multiplication**: Each element of the tensor is multiplied by the corresponding element in the other tensor. No summing is done, just direct element-by-element multiplication.