Chapter_1_TensorBasics

January 25, 2022

1 Getting Sht Done with PyTorch - Venelin Valkov

This will be the notebook which accompanies this book "Getting sht done with PyTorch by Venelin Valkov", in order to take notes as well as test out the code

1.1 Chapter 1: Getting Started with PyTorch

Tensor Basics

[ ]:
```
# import necessary dependencies
import torch
import numpy as np
```

[ ]:
```
# simple numpy arrays
a,b = np.array([5,5,5]), np.array([6,6,6])
# we can add them
c = a + b
print(c) # 5+6 for all the elements

# and now, wow, the same for tensors!!!
a,b = torch.tensor([7,7,7]), torch.tensor([8,8,8])
c = a + b
print(c) # 7+8 for all the elements

# we can even go from numpy to tensors!
print("\nNumpy --> Tensors")
a = torch.tensor([1,1,1])
print(a)
print(type(a)) # tensor type
a = a.numpy()
print(a)
print(type(a)) # numpy nd.array!

# and we can even go from tensors to numpy !!!
print("\nTensors --> Numpy")
a = np.array([9,9,9])
print(a)
print(type(a))
# converting it to a tensor
```

```
a = torch.from_numpy(a)
print(a)
print(type(a))
```

```
[11 11 11]
tensor([15, 15, 15])

Numpy --> Tensors
tensor([1, 1, 1])
<class 'torch.Tensor'>
[1 1 1]
<class 'numpy.ndarray'>

Tensors --> Numpy
[9 9 9]
<class 'numpy.ndarray'>
tensor([9, 9, 9])
<class 'torch.Tensor'>
```

So what are Tensors?!

In short: they are nd-arrays, why nd?, because it is the number of indices required in order to access a specific element! The easiest way to think about it, is that the tensors we use with PyTorch, are basically n-dimensional arrays. This are able to store information in a better sense, because what does it mean to store a video information in a n x d matrix? A tensor gives you more dimensions to describe the data even more!

Creating Tensors

[ ]:
```
# create a tensor with ints and create floats
a = torch.FloatTensor([[1,1,1,1],[2,2,2,2]])
print("This is a float tensor!")
print(a) # we see that there is a period, which indicates the type float

# now we can also define the type inside the tensor operation
a = torch.tensor([[1,2],[2,1]],dtype=torch.bool)
print("\nAnd this is a bool tensor!!")
print(a)


# we can also create tensors with random values - I think range [0,1]
print("\nCreating a tensor with random values")
a = torch.rand(2,2,)
print(a)

# or we can create the tensors with ones
print("\nCreating a tensor with ones")
a = torch.ones(3,2)
print(a)
```

```
This is a float tensor!
tensor([[1., 1., 1., 1.],
        [2., 2., 2., 2.]])

And this is a bool tensor!!
tensor([[True, True],
        [True, True]])

Creating a tensor with random values
tensor([[0.7617, 0.2288],
        [0.4510, 0.2298]])

Creating a tensor with ones
tensor([[1., 1.],
        [1., 1.],
        [1., 1.]])
```

Tensor Operations

[ ]:
```python
# we can get the sum of a tensor
print("Getting the sum of a tensor")
a = torch.tensor([[1,2,3],[4,5,6]])
print(a)
print(f'The sum of the tensor is: {a.sum()}')

# or we can transpose the tensors
print("\nTransposing the tensors")
print(a)
# transposing it
print(a.t())

# we can get the dimensions of the tensors
print("\nDimensions of the tensors")
a = torch.tensor([[3,4,5],[6,7,8]])
print(a)
print("Dimension of the tensor is ",a.size())


# now we can perform mathematical operations
print("\nOperations on tensors")
a = torch.tensor([[1,2,3],[4,5,6]])
b = torch.tensor([[7,8,9],[1,2,3]])
# there are two ways of adding: inplace or not-inplace
c = a.add(b) # not inplace
print(a)
print(b)
print(c)
print("\nInplace operations, modifies the original variable")
```

```
a.add_(b) # now this is the same as doing a + b = c, and then saying c = a.
print(a)
```

```
Getting the sum of a tensor
tensor([[1, 2, 3],
        [4, 5, 6]])
The sum of the tensor is: 21

Transposing the tensors
tensor([[1, 2, 3],
        [4, 5, 6]])
tensor([[1, 4],
        [2, 5],
        [3, 6]])

Dimensions of the tensors
tensor([[3, 4, 5],
        [6, 7, 8]])
Dimension of the tensor is  torch.Size([2, 3])

Operations on tensors
tensor([[1, 2, 3],
        [4, 5, 6]])
tensor([[7, 8, 9],
        [1, 2, 3]])
tensor([[ 8, 10, 12],
        [ 5,  7,  9]])

Inplace operations, modifies the original variable
tensor([[ 8, 10, 12],
        [ 5,  7,  9]])
```