Machine Learning
Pytorch Tutorial 2

Documentation and Common Errors

TA : 許湛然 (Chan-Jan Hsu)
2021.03.05

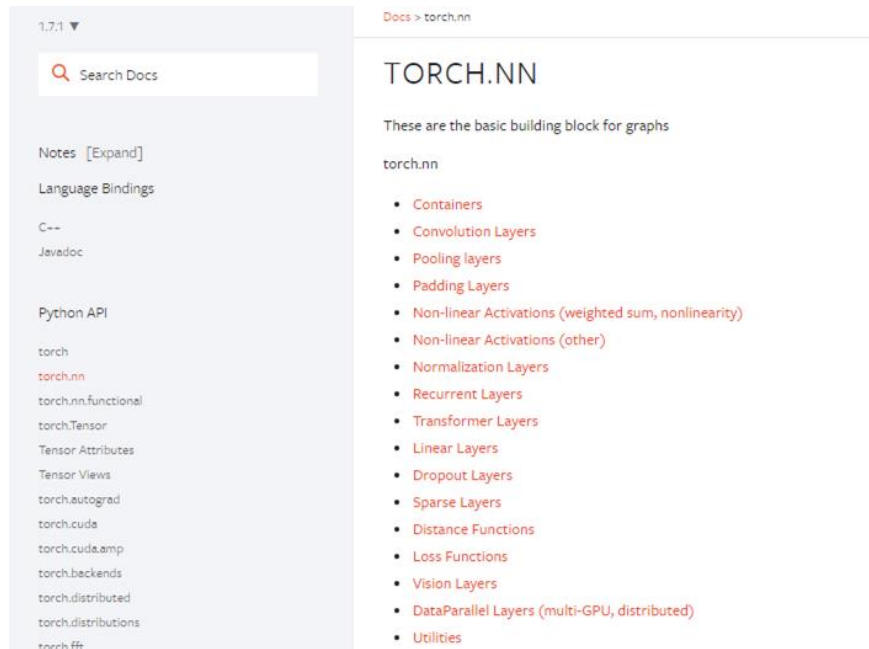# PyTorch Documentation

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

torch.nn -> neural network

torch.optim -> optimization algorithms

torch.utils.data -> dataset, dataloader

PyTorch Documentation Example

function inputs and outputs

data type and explanation
of each input

TORCH.MAX

`torch.max(input) → Tensor`

Returns the maximum value of all elements in the `input` tensor.

WARNING

This function produces deterministic (sub)gradients unlike `max(dim=0)`

Parameters

input (*Tensor*) – the input tensor.

PyTorch Documentation Example

Some functions behave differently with different inputs

Parameters : You don't need to specify the name of the argument (Positional Arguments)

Keyword Arguments : You have to specify the name of the argument

*They are separated by ***

torch.max(*input, dim, keepdim=False, *, out=None*) -> *(Tensor, LongTensor)*

Returns a namedtuple `(values, indices)` where `values` is the maximum value of each row of the `input` tensor in the given dimension `dim`. And `indices` is the index location of each maximum value found (argmax).

If `keepdim` is `True`, the output tensors are of the same size as `input` except in the dimension `dim` where they are of size 1. Otherwise, `dim` is squeezed (see `torch.squeeze()`), resulting in the output tensors having 1 fewer dimension than `input`.

● NOTE

If there are multiple maximal values in a reduced row then the indices of the first maximal value are returned.

Parameters

- **input** (*Tensor*) – the input tensor.
- **dim** (*int*) – the dimension to reduce.
- **keepdim** (*bool*) – whether the output tensor has `dim` retained or not. Default: `False`.

Keyword Arguments

- **out** (*tuple, optional*) – the result tuple of two output tensors (max, max_indices)

PyTorch Documentation Example

Some functions behave differently with different inputs

Arguments with default value : Some arguments have a default value (keepdim=False), so passing a value of this argument is optional

```
torch.max(input, dim, keepdim=False, *, out=None) -> (Tensor, LongTensor)
```

Returns a namedtuple (values, indices) where values is the maximum value of each row of the input tensor in the given dimension dim. And indices is the index location of each maximum value found (argmax).

If keepdim is True, the output tensors are of the same size as input except in the dimension dim where they are of size 1. Otherwise, dim is squeezed (see torch.squeeze()), resulting in the output tensors having 1 fewer dimension than input.

> • NOTE
>
> If there are multiple maximal values in a reduced row then the indices of the first maximal value are returned.

Parameters

- **input** (*Tensor*) – the input tensor.
- **dim** (*int*) – the dimension to reduce.
- **keepdim** (*bool*) – whether the output tensor has dim retained or not. Default: False.

Keyword Arguments

- **out** (*tuple*, *optional*) – the result tuple of two output tensors (max, max_indices)

PyTorch Documentation Example

Three Kinds of torch.max

1. `torch.max(input) → Tensor`
2. `torch.max(input, dim, keepdim=False, *, out=None) → (Tensor, LongTensor)`
3. `torch.max(input, other, *, out=None) → Tensor`

input : Tensor, dim : int, keepdim : bool
other : Tensor

PyTorch Documentation Example

1.torch.max(input) → Tensor

Find the maximum value of a tensor, and return that value.

input

[[1   2   3]

 [5   6   4]]

PyTorch Documentation Example

2. `torch.max(input, dim, keepdim=False, *, out=None)` → `(Tensor, LongTensor)`

Find the maximum value of a tensor along a dimension, and return that value, along with the index corresponding to that value.

input

[[1 2 7]

[5 6 4]]

PyTorch Documentation Example

3.torch.max(input, other) → Tensor

Perform element-wise comparison between two tensors of the same size, and select the maximum of the two to construct a tensor with the same size.

input

[[1   2   3]   [[2   4   6]

[5   6   4]]   [1   3   5]]

PyTorch Documentation Example (Colab)

Three Kinds of torch.max

1. `torch.max(input) → Tensor`
2. `torch.max(input, dim, keepdim=False, *, out=None) → (Tensor, LongTensor)`
3. `torch.max(input, other, *, out=None) → Tensor`

```
input : Tensor
dim : int
keepdim : bool
other : Tensor
```

Colab code

```
x = torch.randn(4,5)
y = torch.randn(4,5)
```

1. `m = torch.max(x)`
2. `m, idx = torch.max(x,0)`→O
   `m, idx = torch.max(input = x,dim=0)`→O
   `m, idx = torch.max(x,0,False)`→O
   `m, idx = torch.max(x,0,keepdim=True)`→O
   `m, idx = torch.max(x,0,False,out=p)`→O
   `m, idx = torch.max(x,0,False,p)`→x
   *out is a keyword argument
   `m, idx = torch.max(x,True)`→x
   *did not specify dim
1. `t = torch.max(x,y)`

Common Errors -- Tensor on Different Device to Model

```
model = torch.nn.Linear(5,1).to("cuda:0")
x = torch.Tensor([1,2,3,4,5]).to("cpu")
y = model(x)
```
Tensor for * is on CPU, but expected them to be on GPU

=> send the tensor to GPU

```
x = torch.Tensor([1,2,3,4,5]).to("cuda:0")
y = model(x)
print(y.shape)
```

Common Errors -- Mismatched Dimensions

```
x = torch.randn(4,5)
y = torch.randn(5,4)
z = x + y
```
The size of tensor a (5) must match the size of tensor b (4) at non-singleton dimension 1

=> the shape of a tensor is incorrect, use **transpose, squeeze, unsqueeze** to align the dimensions

```
y = y.transpose(0,1)
z = x + y
print(z.shape)
```

Common Errors -- Cuda Out of Memory

```python
import torch
import torchvision.models as models
resnet18 = models.resnet18().to("cuda:0") # Neural Networks for Image Recognition
data = torch.randn(512,3,244,244) # Create fake data (512 images)
out = resnet18(data.to("cuda:0")) # Use Data as Input and Feed to Model
print(out.shape)
```

CUDA out of memory. Tried to allocate 350.00 MiB (GPU 0; 14.76 GiB total capacity; 11.94 GiB already allocated; 123.75 MiB free; 13.71 GiB reserved in total by PyTorch)

=> The batch size of data is too large to fit in the GPU. Reduce the batch size.

Common Errors -- Cuda Out of Memory

If the data is iterated (batch size = 1), the problem will be solved. You can also use DataLoader

```python
for d in data:
 out = resnet18(d.to("cuda:0").unsqueeze(0))
print(out.shape)
```

Common Errors -- Mismatched Tensor Type

```python
import torch.nn as nn
L = nn.CrossEntropyLoss()
outs = torch.randn(5,5)
labels = torch.Tensor([1,2,3,4,0])
lossval = L(outs,labels)  # Calculate CrossEntropyLoss between outs and labels
```

expected scalar type Long but found Float

=> labels must be long tensors, cast it to type "Long" to fix this issue

```python
labels = labels.long()
lossval = L(outs,labels)
print(lossval)
```