PyTorch Fundamentals - Gradients

Tensors with Gradients

Creating Tensors with Gradients

• Allows accumulation of gradients

```
Method 1: Create tensor with gradients

It is very similar to creating a tensor, all you need to do is to add an additional argument.

import torch

a = torch.ones((2, 2), requires_grad=True)
a
```

Check if tensor requires gradients

This should return True otherwise you've not done it right.

a.requires_grad

True

1

Method 2: Create tensor with gradients

This allows you to create a tensor as usual then an additional line to allow it to accumulate gradients.

```
# Normal way of creating gradients
a = torch.ones((2, 2))

# Requires gradient
a.requires_grad_()

# Check if requires gradient
a.requires_grad
```

True



A tensor without gradients just for comparison

If you do not do either of the methods above, you'll realize you will get False for checking for gradients.

```
# Not a variable
no_gradient = torch.ones(2, 2)
no_gradient.requires_grad
```

False

Tensor with gradients addition operation

```
# Behaves similarly to tensors
b = torch.ones((2, 2), requires_grad=True)
print(a + b)
print(torch.add(a, b))
```

Tensor with gradients multiplication operation

As usual, the operations we learnt previously for tensors apply for tensors with gradients. Feel free to try divisions, mean or standard deviation!

```
print(a * b)
print(torch.mul(a, b))
```

Manually and Automatically Calculating Gradients

What exactly is requires_grad ? - Allows calculation of gradients w.r.t. the tensor that all allows gradients accumulation

$$y_i = 5(x_i+1)^2$$

Create tensor of size 2x1 filled with 1's that requires gradient

```
x = torch.ones(2, requires_grad=True)
x
```

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

Simple linear equation with x tensor created

$$\left. y_i \right|_{x_i=1} = 5(1+1)^2 = 5(2)^2 = 5(4) = 20$$

We should get a value of 20 by replicating this simple equation

```
y = 5 * (x + 1) ** 2
```

tensor([20., 20.])

Simple equation with y tensor

Backward should be called only on a scalar (i.e. 1-element tensor) or with gradient w.r.t. the variable

Let's reduce y to a scalar then...

$$o=rac{1}{2}\sum_i y_i$$

As you can see above, we've a tensor filled with 20's, so average them would return 20

$$o = (1/2) * torch.sum(y)$$

o

tensor(20.)

Calculating first derivative

Recap y equation: $y_i = 5(x_i+1)^2$

Recap o equation: $o=rac{1}{2}\sum_i y_i$

Substitute y into o equation: $o=rac{1}{2}\sum_i 5(x_i+1)^2$

$$rac{\partial o}{\partial x_i} = rac{1}{2}[10(x_i+1)]$$

$$rac{\partial o}{\partial x_i}ig|_{x_i=1} = rac{1}{2}[10(1+1)] = rac{10}{2}(2) = 10$$

We should expect to get 10, and it's so simple to do this with PyTorch with the following line...

Get first derivative:

o.backward()

Print out first derivative:

x.grad

tensor([10., 10.])

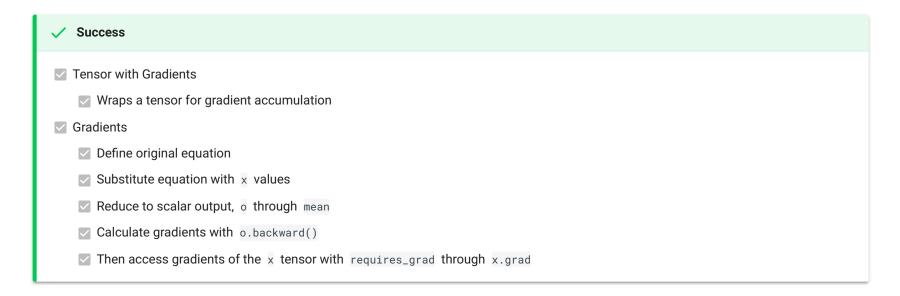
If x requires gradient and you create new objects with it, you get all gradients

print(x.requires_grad)
print(y.requires_grad)
print(o.requires_grad)

True True True

Summary

We've learnt to...



Comments

Deep Learning Wizard Comment Policy

