```
In [1]: import torch
```

```
In [2]: x = torch.arange(4.0)
        x
```

```
Out[2]: tensor([0., 1., 2., 3.])
```

```
In [3]: x.requires_grad_(True)
        x.grad
```

```
In [4]: y = 2 * torch.dot(x,x)
        y
```

```
Out[4]: tensor(28., grad_fn=<MulBackward0>)
```

```
In [5]: y.backward()
        x.grad
```

```
Out[5]: tensor([ 0.,  4.,  8., 12.])
```

```
In [6]: x.grad == 4*x
```

```
Out[6]: tensor([True, True, True, True])
```

```
In [7]: x.grad.zero_()
        y = x.sum()
        y.backward()
        x.grad
```

```
Out[7]: tensor([1., 1., 1., 1.])
```

```
In [8]: x.grad.zero_()
        y = x*x
        y.backward(gradient=torch.ones(len(y)))
        x.grad
```

```
Out[8]: tensor([0., 2., 4., 6.])
```

```
In [9]: x.grad.zero_()
        y = x * x
        u = y.detach()
        z = u * x

        z.sum().backward()
        x.grad == u
```

```
Out[9]: tensor([True, True, True, True])
```

```
In [10]: x.grad.zero_()
         y.sum().backward()
         x.grad == 2 * x
```

```
Out[10]: tensor([True, True, True, True])
```

```
In [11]: def f(a):
             b = a * 2
             while b.norm() < 1000:
                 b = b * 2
             if b.sum() > 0:
                 c = b
             else:
                 c = 100 * b
             return c
```

```
In [12]: a = torch.randn(size=(), requires_grad=True)
         d = f(a)
         d.backward()
```

```
In [13]: a.grad == d / a
```

```
Out[13]: tensor(True)
```

Discussion: This unit covers basic concepts related to automatic differentiation, but compared to the previous sections, I found it a bit challenging to understand the code at first. Perhaps it's because I skipped section 2_4, and going through that might have made this one easier to grasp. Nevertheless, I found it fascinating how we can create y by taking the dot product of a tensor with itself and multiplying it by 2 (in this case, y is a scalar value), then use the .backward() function to compute the gradients and check the results. Instead of trying to fully grasp everything in this unit, my goal is to understand what .backward() and .grad mean, as well as remember that .grad.zero_()

serves to reset the gradients.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js