Week 2 Questions (220962018)

January 11, 2025

0.1 Week 2 Lab Exercise

[15]:
```
import torch
device = "cuda" if torch.cuda.is_available() else "cpu"
torch.cuda.device_count() , device
```

[15]:
```
(1, 'cuda')
```

0.2 Lab Exercise

0.2.1 1) Draw Computation Graph and work out the gradient dz/da by following the path back from z to a and compare the result with the analytical gradient.

$$x = 2a + 3b$$

$$y = 5a^2 + 3b^3$$

$$z = 2x + 3y$$

[16]:
```
a = torch.tensor(1.0, requires_grad=True)
b = torch.tensor(1.0, requires_grad=True)

x = 2*a + 3*b
y = 5*a**2 + 3*b**3
z = 2*x + 3*y

z.backward()

print(f"Gradient of z with respect to a: {a.grad.item()}")

print("Analytical Solution: ",(4+ 30 * a).item())
```

```
Gradient of z with respect to a: 34.0
Analytical Solution:  34.0
```

0.2.2 2) For the following Computation Graph, work out the gradient da/dw by following the path back from a to w and compare the result with the analytical gradient.

[17]:
```
x = torch.tensor(1.0, requires_grad=True)
b = torch.tensor(1.0, requires_grad=True)
w = torch.tensor(1.0, requires_grad=True)

u = w * x
v = u + b
a = torch.relu(v)

a.backward()

print(f"Gradient of a with respect to w: {w.grad.item()}")
```

```
Gradient of a with respect to w: 1.0
```

0.2.3 3) Repeat the Problem 2 using Sigmoid function

[18]:
```
x = torch.tensor(1.0, requires_grad=True)
b = torch.tensor(1.0, requires_grad=True)
w = torch.tensor(1.0, requires_grad=True)

u = w * x
v = u + b
a = torch.sigmoid(v)

a.backward()

print(f"Gradient of a with respect to w: {w.grad.item()}")

print("Analytical Solution: ",(a*(1-a)*x).item())
```

```
Gradient of a with respect to w: 0.10499362647533417
Analytical Solution:  0.10499362647533417
```

0.2.4 4) Verify that the gradients provided by PyTorch match with the analytical gradients of the function f= exp(-x2-2x-sin(x)) w.r.t x

[19]:
```
x = torch.tensor(1.0, requires_grad=True)

u = x**2
v = 2*x
w = torch.sin(x)
f = torch.exp(-u - v - w)

f.backward()
```

```
print(f"Gradient of f with respect to x: {x.grad.item()}")
```

```
Gradient of f with respect to x: -0.09744400531053543
```

0.2.5 5) Compute gradient for the function y=8x4+ 3x3 +7x2+6x+3 and verify the gradients provided by PyTorch with the analytical gradients. A snapshot of the Python code is provided below.

[20]:
```
x=torch.tensor(2.0, requires_grad=True)

y=8*x**4+3*x**3+7*x**2+6*x+3
y.backward()

print(f"Gradient of y with respect to x: {x.grad.item()}")

print("Analytical Solution: ",(32*x**3+9*x**2+14*x+6).item())
```

```
Gradient of y with respect to x: 326.0
Analytical Solution:  326.0
```

0.2.6 Calculate the intermediate variables a,b,c,d, and e in the forward pass. Starting from f, calculate the gradient of each expression in the backward pass manually. Calculate f/ y using the computational graph and chain rule. Use the chain rule to calculate gradient and compare with analytical gradient.

[21]:
```
x = torch.tensor(1.0, requires_grad=True)
y = torch.tensor(1.0, requires_grad=True)
z = torch.tensor(1.0, requires_grad=True)

b = torch.sin(y)
print(f"b (sin(y)): {b.item()}")

a = 2 * x
print(f"a (2 * x): {a.item()}")

c = a / b
print(f"c (a / b): {c.item()}")

d = c * z
print(f"d (c * z): {d.item()}")

e = torch.log(d + 1)
print(f"e (log(d + 1)): {e.item()}")

f = torch.tanh(e)
print(f"f (tanh(e)): {f.item()}")
```

```python
f.backward()


print(f"Gradient of f with respect to y: {y.grad.item()}")

print("Analytical Solution: ",((1-torch.tanh(e)**2)*(1/(d+1))*(-(a/b**2))*torch.
 ↪cos(y)).item())
```

```
b (sin(y)): 0.8414709568023682
a (2 * x): 2.0
c (a / b): 2.3767902851104736
d (c * z): 2.3767902851104736
e (log(d + 1)): 1.2169256210327148
f (tanh(e)): 0.8387449383735657
Gradient of f with respect to y: -0.13400448858737946
Analytical Solution:  -0.13400450348854065
```