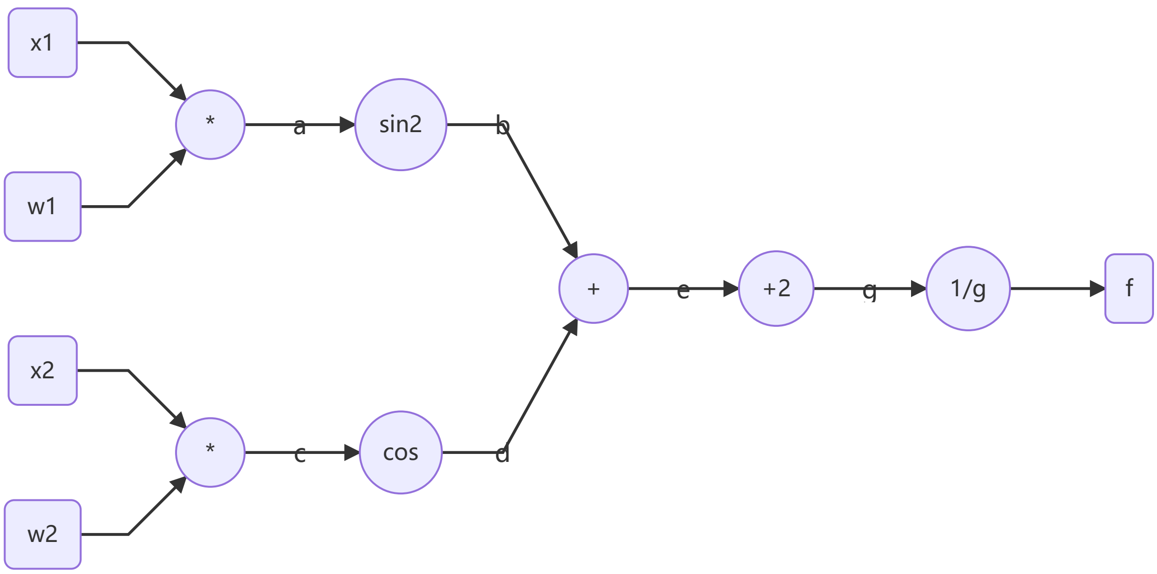
HomeWork 2

Q1

(a)

Computational graph:



Where we have:

$$
\begin{aligned}
f(g) = \frac{1}{g} &\longrightarrow \frac{\mathrm{d}f}{\mathrm{d}g} = \frac{-1}{g^2} \\
g(e) = e + 2 &\longrightarrow \frac{\mathrm{d}g}{\mathrm{d}e} = 1 \\
e(b,d) = b + d &\longrightarrow \frac{\partial e}{\partial b} = \frac{\partial e}{\partial d} = 1 \\
b(a) = sin^2 a &\longrightarrow \frac{\mathrm{d}b}{\mathrm{d}a} = sin2a \\
a(x_1, w_1) = x_1 w_1 &\longrightarrow \frac{\mathrm{d}a}{\mathrm{d}x_1} = w_1, \frac{\mathrm{d}a}{\mathrm{d}w_1} = x_1 \\
d(c) = cosc &\longrightarrow \frac{\mathrm{d}d}{\mathrm{d}c} = -sinc \\
c(x_2, w_2) = x_2 w_2 &\longrightarrow \frac{\mathrm{d}c}{\mathrm{d}x_2} = w_2, \frac{\mathrm{d}c}{\mathrm{d}w_2} = x_2
\end{aligned}
$$

So that:

$$
\begin{aligned}
\frac{\partial f}{\partial x_1} &= \frac{-w_1 sin(2x_1 w_1)}{(2 + sin^2(x_1 w_1) + cos(x_2 w_2))^2} \\
\frac{\partial f}{\partial w_1} &= \frac{-x_1 sin(2x_1 w_1)}{(2 + sin^2(x_1 w_1) + cos(x_2 w_2))^2} \\
\frac{\partial f}{\partial x_2} &= \frac{w_2 sin(x_2 w_2)}{(2 + sin^2(x_1 w_1) + cos(x_2 w_2))^2} \\
\frac{\partial f}{\partial w_2} &= \frac{x_2 sin(x_2 w_2)}{(2 + sin^2(x_1 w_1) + cos(x_2 w_2))^2}
\end{aligned}
$$

(b)

Implementation

Numerical:

```
def numericalGradient(x1, w1, x2, w2):
    deno = (2 + sin(x1 * w1) ** 2 + cos(x2*w2)) ** 2
    n_x1 = -1*w1*sin(2*x1*w1)
    n_w1 = -1*x1*sin(2*x1*w1)
    n_x2 = w2*sin(x2*w2)
    n_w2 = x2*sin(x2*w2)
    return n_x1/deno, n_w1/deno, n_x2/deno, n_w2/deno
```

computational graph:

```
forward = [
    lambda args: (args[0] * args[1], args[2] * args[3]),
    lambda args: (sin(args[0]) ** 2, cos(args[1])),
    lambda args: sum(args),
    lambda arg: arg + 2,
    # lambda arg: 1/arg
]
backward = [
    lambda arg, up:  up * -1 / (arg ** 2),
    lambda args, up: up,
    lambda args, up: (up * 1, up * 1),
    lambda args, up: (up[0]*sin(2*args[0]), up[1] * -sin(args[1])),
    lambda args, up: (up[0]*args[1], up[0] * args[0],up[1]*args[3],
up[1]*args[2])
]
def cgGradient(x1, w1, x2, w2):
    mid = []
    args = (x1, w1, x2, w2)
    mid.append(args)
    for func in forward:
        temp = func(args)
        args = temp
        mid.append(temp)
    up = 1
    for func in backward:
        args = mid.pop()
        up = func(args, up)
    return up
```

Test

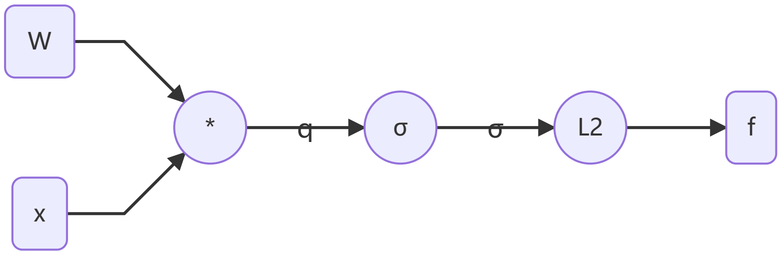
```
>print(numericalGradient(1, 2, 3, 4))
 print(cgGradient(1, 2, 3, 4))
>(0.11233639973639942, 0.05616819986819971, -0.15929299963446267,
-0.11946974972584701)
 (0.1123363997363994, 0.0561681998681997, -0.15929299963446267,
-0.11946974972584701)
```

Verified !

Q2

(a)

Computational graph:



Where we have:

$$
\begin{aligned}
f(\boldsymbol{m}) = \sum_i \sigma_i^2 &\longrightarrow \frac{\partial f}{\partial \sigma_i} = 2\sigma_i \\
\sigma(q_i) = \frac{1}{1+e^{-q_i}} &\longrightarrow \frac{\mathrm{d}\sigma(q_i)}{\mathrm{d}q_i} = (1-\sigma(q_i)\sigma(q_i) \\
q = \boldsymbol{W}\boldsymbol{x} &\longrightarrow \frac{\partial q_k}{\partial W_{ij}} = 1_{k=i} x_j, \frac{\partial q_k}{\partial x_i} = W_{ki}
\end{aligned}
$$

So that:

$$
\begin{aligned}
\nabla_W f &= 2(1-\sigma(\boldsymbol{q})\sigma^2(\boldsymbol{q}) \cdot \boldsymbol{x}^T \\
\nabla_x f &= 2\boldsymbol{W}^T \cdot (1-\sigma(\boldsymbol{q})\sigma^2(\boldsymbol{q})
\end{aligned}
$$

(b)

Implementation

Numerical:

```
def numericalGradient(x, w):
    q = w.dot(x)
    sig = sigmod(q)
    return np.dot(w.T, (2*(1-sig)*sig**2)), np.dot((2*(1-sig)*sig**2), x.T)
```

Computational graph:

```
forward = [
    lambda args: args[1].dot(args[0]),
    lambda arg: sigmod(arg),
    # lambda arg: sum([q**2 for q in arg])
]
backward = [
    lambda arg, up: up * 2*arg,
    lambda arg, up: up*(1-sigmod(arg))*sigmod(arg),
    lambda args, up: (args[1].T.dot(up), up.dot(args[0].T))
]
def cgGradient(x,w) # same as above
```

Test

```
>w = np.array([
    [-1, -2, -3],
    [2, 3, 4],
    [3, 4, 5]
])
x = np.array([-1, 3, 2])[:, None]
nG = numericalGradient(x, w)
cG = cgGradient(x, w)
print((nG[0] == cG[0]).all())
> True
```

Verified !