202001555 지은미

In [15]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
import numpy as np

#set some inputs
x = -1; y = 5; z = -4

#perform the forward pass
q=x+y # q -> 3
f=q*z # f -> -12

# perfom the backward pass(backpropagation) in reverse order
#first backprop through f=q*z
dfdz=q
dfdq=z
# now backprop through q = x+y
dfdx=1.0 * dfdz
dfdy=1.0 * dfdq

print('q =', q)
print('f =', f)
print('dfdq =',dfdq)
print('dfdx =',dfdx)
print('dfdy =',dfdy)
print('dfdz =',dfdz)
```

```
q = 4
f = -16
dfdq = -4
dfdx = 4.0
dfdy = -4.0
dfdz = 4
```

In [16]:
```
import math

w=[2,-3,-3]
x=[-1,-2]

#forward pass
dot = w[0]*x[0] + w[1]*x[1] + w[2]
f=1.0/(1+math.exp(-dot))
print(dot)
print(f)

ddot = (1-f)*f
dx=[w[0]*ddot, w[1]*ddot]
dw=[x[0]*ddot, x[1]*ddot,1.0*ddot]
print(ddot)
print(dx)
print(dw)
```

```
1
0.7310585786300049
0.19661193324148185
[0.3932238664829637, -0.5898357997244456]
[-0.19661193324148185, -0.3932238664829637, 0.19661193324148185]
```

In [17]:
```
x = 3 # example values
y = -4

# forward pass
sigy = 1.0 / (1 + math.exp(-y)) # sigmoid in numerator
num = x + sigy # numerator
```

```
sigx = 1.0 / (1 + math.exp(-x)) # sigmoid in denominator
xpy = x + y
xpysqr = xpy**2
den = sigx + xpysqr # denominator
invden = 1.0 / den
f = num * invden
```

In [26]:
```
# backprop f = num * invden
dnum = invden # gradient on numerator
dinvden = num
# backprop invden = 1.0 / den
dden = (-1.0 / (den**2)) * dinvden
# backprop den = sigx + xpysqr
dsigx = (1) * dden
dxpysqr = (1) * dden
# backprop xpysqr = xpy**2
dxpy = (2 * xpy) * dxpysqr
# backprop xpy = x + y
dx = (1) * dxpy
dy = (1) * dxpy
# backprop sigx = 1.0 / (1 + math.exp(-x))
dx += ((1 - sigx) * sigx) * dsigx # Notice += !! See notes below
# backprop num = x + sigy
dx += (1) * dnum
dsigy = (1) * dnum
# backprop sigy = 1.0 / (1 + math.exp(-y))
dy += ((1 - sigy) * sigy) * dsigy

print(dx)
print(dx)
```

```
2.0595697955721652
2.0595697955721652
```

In [27]:
```
# forward pass
W = np.random.randn(5, 10)
X = np.random.randn(10, 3)
D = W.dot(X)

# now suppose we had the gradient on D from above in the circuit
dD = np.random.randn(*D.shape) # same shape as D
dW = dD.dot(X.T) #.T gives the transpose of the matrix
dX = W.T.dot(dD)

print(dD)
print(dW)
print(dX)
```

```
[[-0.57198597  0.89584679  0.84204558]
 [-0.32521243 -2.04828155 -0.77450263]
 [ 1.4745193  -0.65705192 -0.95865839]
 [-0.61682902  0.66229266  0.87706421]
 [ 0.68051829 -0.37271381  1.08354016]]
[[ 0.93010259  0.46345812 -0.20389477 -1.08937897 -0.94169119  1.0698664
   0.41523306 -0.90507069 -1.3414823   0.42046197]
 [-1.29062718 -0.58995585 -1.15319626  2.43116977  1.65752167 -3.77852646
  -0.47733384  2.11848973  1.42078012 -1.89586057]
 [-0.63184931 -0.5287229   0.41379144  0.41940127  0.58754645  0.24218392
  -0.24697282  0.25208158  1.72303525  1.05433202]
 [ 0.95984907  0.43200235 -0.57212514 -0.9349168  -0.87903659  0.68134299
   0.46924391 -0.77458455 -1.28235903  0.42216016]
 [ 1.94795898  0.17728533 -2.86880171 -1.34455624 -1.31037899  0.63658626
   1.27223868 -1.28539761 -0.40363735  3.37115341]]
[[-2.32751552  3.08952414  5.18263769]
 [-2.04396579  1.25170815  4.58009198]
 [ 2.21772683 -0.65395437 -0.69794183]
```

```
[-0.85206765 -2.5579467   0.93588596]
[-1.02791356  0.34415677  1.80187732]
[-0.96650754  4.02374918  1.23716799]
[ 0.78720639  1.02475976 -0.64091406]
[-0.88957683  0.45234883  1.3599559 ]
[ 1.24491799 -0.11426234  3.18707446]
[ 1.06786745  1.85323924  0.89561264]]
```

lab.9

In [20]:
```
# f(x,y) = xy(x=4, y=-3)

x=4
y=-3

f=x*y

dfdx=y  # y=-3
dfdy=x  #x=4

print('f =', f)
print('dfdx =',dfdx)
print('dfdy =',dfdy)
```

```
f = -12
dfdx = -3
dfdy = 4
```

In [21]:
```
#f(x,y)=x+y

f=x+y

dfdx=1
dfdy=1

print('f =', f)
print('dfdx =',dfdx)
print('dfdy =',dfdy)
```

```
f = 1
dfdx = 1
dfdy = 1
```

In [22]:
```
#f(x,y)=max(x,y)

f=max(x,y) #f=x

if(x>=y):
    dfdx=1
    dfdy=0
else:
    dfdx=0
    dfdy=1

print('f =', f)
print('dfdx =',dfdx)
print('dfdy =',dfdy)
```

```
f = 4
dfdx = 1
dfdy = 0
```

In [23]:
```
#f(x,y,z)=(x+y)*z (x=-2, y=5, z=-4)

x=-2
y=5
z=-4
```

```
q=x+y
f=q*z

dqdx = 1
dqdy = 1
dfdq = z  #-4
dfdz = q  #-2+5 = 3

dfdx=1.0*dfdq
dfdy=1.0*dfdq

print('q =', q)
print('f =', f)
print('dfdq =',dfdq)
print('dfdx =',dfdx)
print('dfdy =',dfdy)
print('dfdz =',dfdz)
```

```
q = 3
f = -12
dfdq = -4
dfdx = -4.0
dfdy = -4.0
dfdz = 3
```

In [25]:
```
# 5번, 시그모이드 예제

w=[2,-3,-3]
x=[-1,-2]

#forward pass
dot=w[0]*x[0]+w[1]*x[1]+w[2]
f=1.0/(1+math.exp(-dot))
print(dot)
print(f)

# backwardpass through the neuron(backpropation)
ddot=(1-f)*f
dx=[w[0]*ddot, w[1]*ddot]
dw=[x[0]*ddot, x[1]*ddot, 1.0*ddot]

print(ddot)
print(dx)
print(dw)
```

```
1
0.7310585786300049
0.19661193324148185
[0.3932238664829637, -0.5898357997244456]
[-0.19661193324148185, -0.3932238664829637, 0.19661193324148185]
```

202001555 지은미