Backpropagation

Most of the materials are taken from here

Nastaran Okati

Outline

- Introduction
- Interpreting of the gradient
- Chain rule
- Modularity
- Gradient for vectorized operations

1.
Introduction

Backpropagation

- A way of computing gradients of expressions through recursive application of chain rule.

- Problem statement: We are given some function f(x) where x is a vector of inputs and we are interested in computing the gradient of f at x (i.e. $\nabla f(x)$).

- We are interested in this problem for the case of Neural Networks.

Interpreting the gradient

The partial derivative for either input:

$$f(x,y) = xy \qquad \rightarrow \qquad \frac{\partial f}{\partial x} = y \qquad \frac{\partial f}{\partial y} = x$$

The derivative tells you the rate of change of a function with respect to that variable surrounding a very small region near a particular point

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- When h is very small, then the function is approximated by a straight line, and the derivative is its slope

- The derivative on each variable tells you the sensitivity of the whole expression on its value.

- Consider a case where x=4 and y=-3. Compute the gradient for f(x,y) = xy with respect to x and y and interpret it.

- ∇ f is the vector of partial derivatives:

$$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}] = [y, x]$$

Derive the derivative for f(x,y) = x + y and f(x,y) = max(x,y) and interpret them.

Chain rule

Consider the function: f(x,y,z) = (x+y)z

q = x+y

f = qz

Based on the previous slides:

$$\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

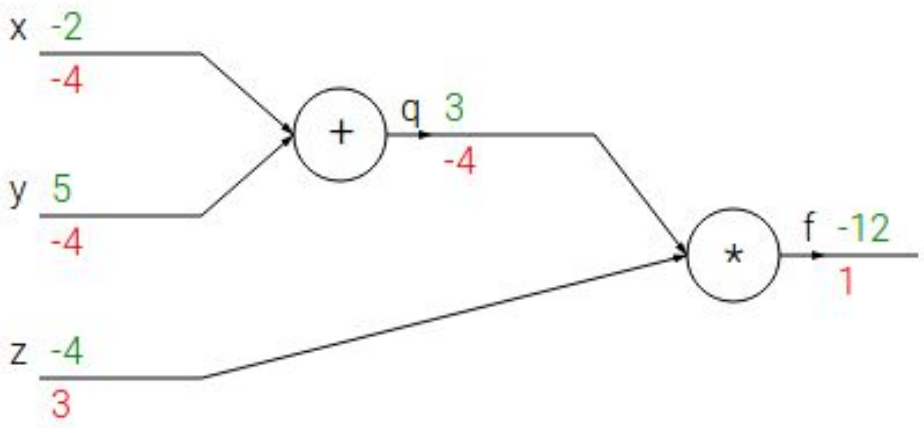
Chain rule

Remember that we are not interested in the gradient of q. We want the gradient of f with respect to x,y,z.

According to the chain rule: $$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

Forward and backward pass

Every gate in a circuit diagram gets some inputs and can right away compute two things: 1. its output value and 2. the local gradient of its inputs with respect to its output value.



- Gates can do this completely independently without being aware of any of the details of the full circuit that they are embedded in.

- Once the forward pass is over, during backpropagation the gate will eventually learn about the gradient of its output value on the final output of the entire circuit.

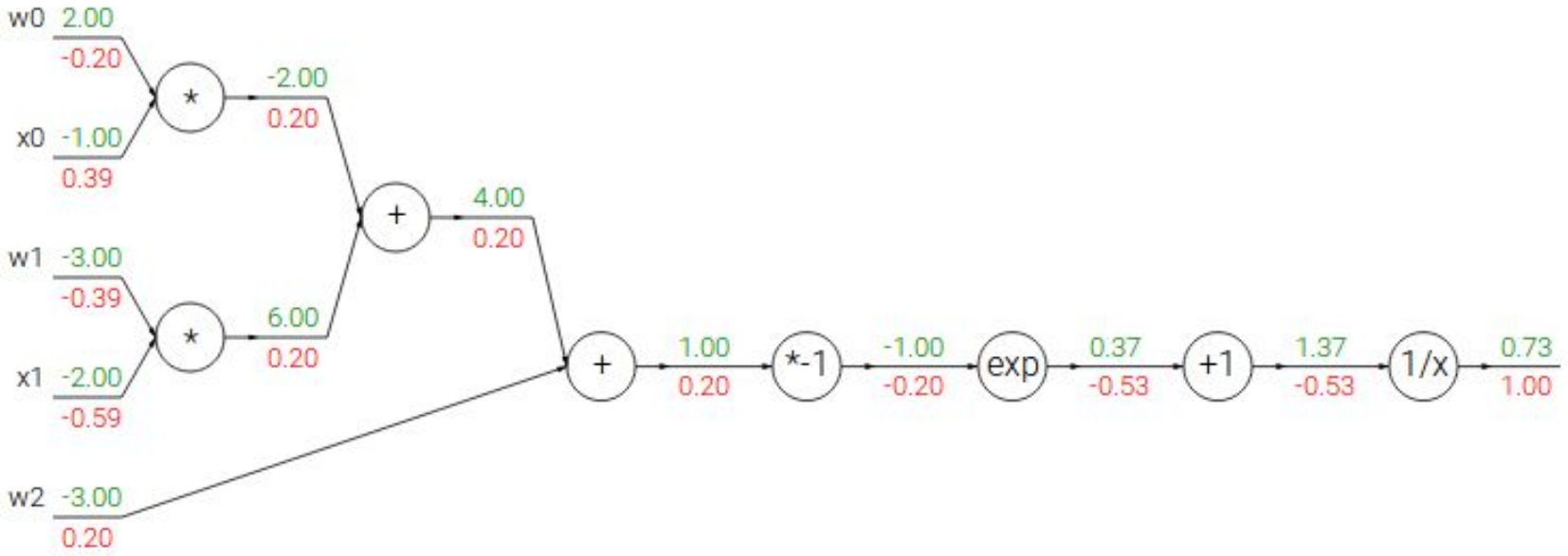
Sigmoid Example

Note that any kind of differentiable function can act as a gate.

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

We can group multiple gates into a single gate, or decompose a function into multiple gates whenever it is convenient.

Sigmoid Example



Exercise

Write the forward and backward pass of the following function:

$$f(x, y) = \frac{x + \sigma(y)}{\sigma(x) + (x + y)^2}$$

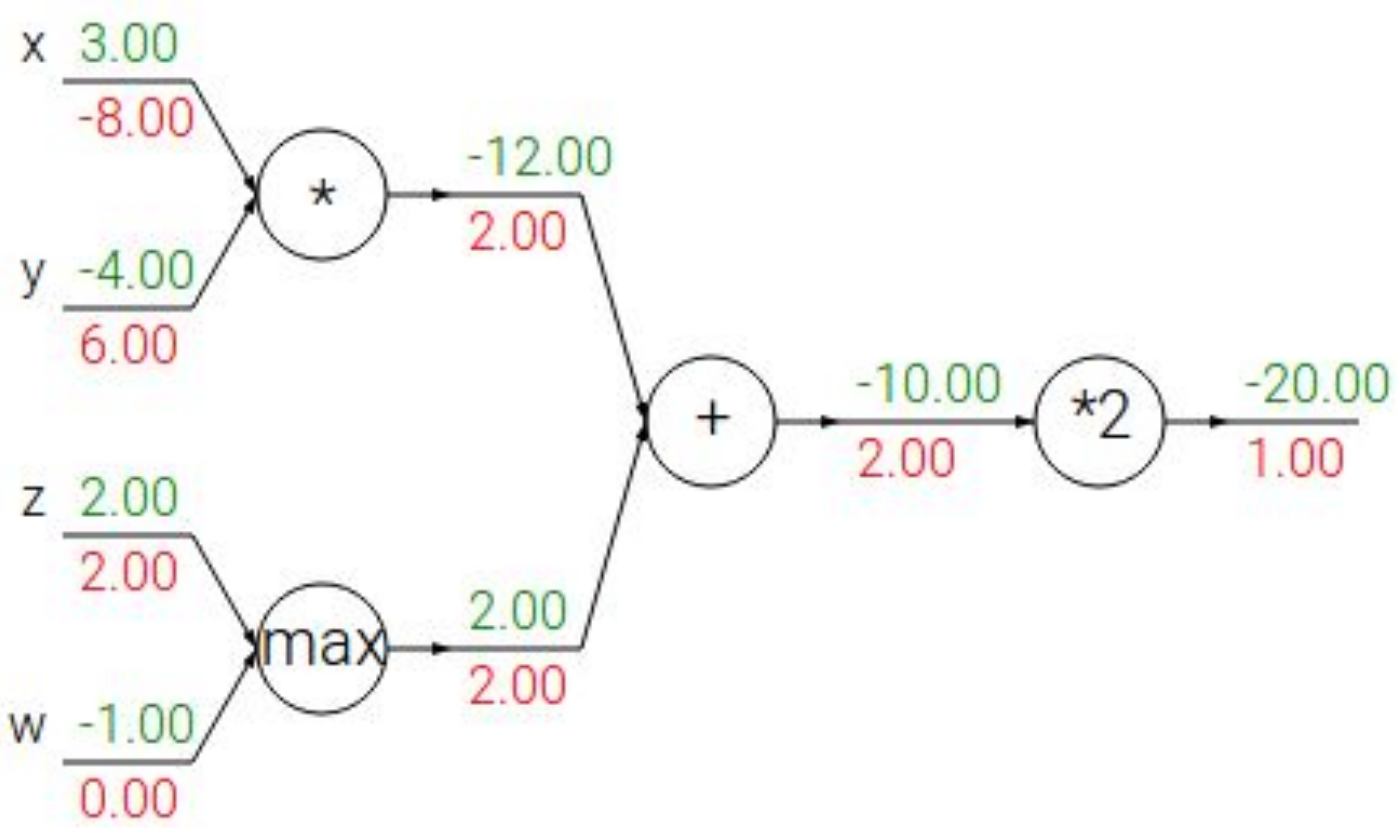
- Cash forward variables!
- Gradients add up at forks!

Patterns in backward flow

- In many cases the backward-flowing gradient can be interpreted on an intuitive level

- The three most commonly used gates in neural networks (add,mul,max), all have very simple interpretations in terms of how they act during backpropagation

Example


x
3.00
-8.00
y
-4.00
6.00
*
-12.00
2.00
z
2.00
2.00
w
-1.00
0.00
max
2.00
2.00
+
-10.00
2.00
*2
-20.00
1.00

- The add gate always takes the gradient on its output and distributes it equally to all of its inputs
- The max gate routes the gradient. It distributes the gradient to exactly one of its inputs
- In the multiply gate its local gradients are the input values (except switched), and this is multiplied by the gradient on its output during the chain rule.

Ready to train Neural Networks?

Wait until next session! 

Thanks!

Any questions?