Backpropagation

By Shrishty Chandra

What is Backpropagation

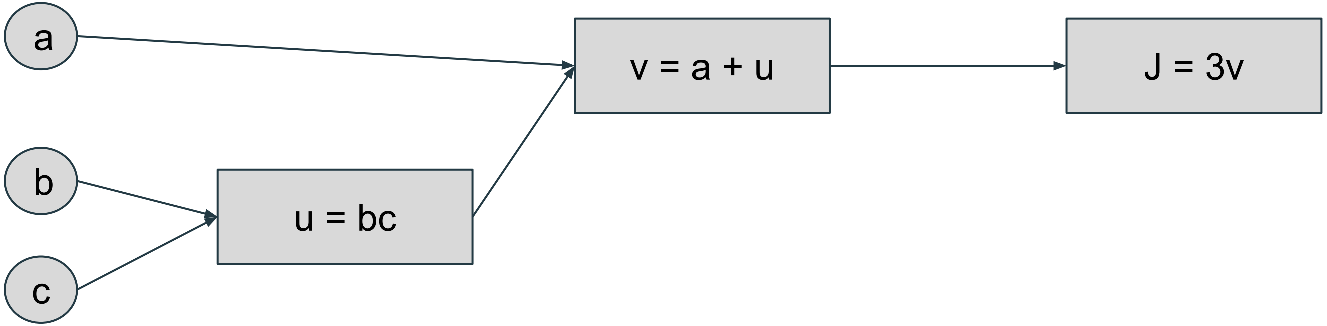
Backpropagation is a method used in artificial neural networks to calculate the error contribution of each neuron after a batch of data

- Wikipedia

Computation Graph

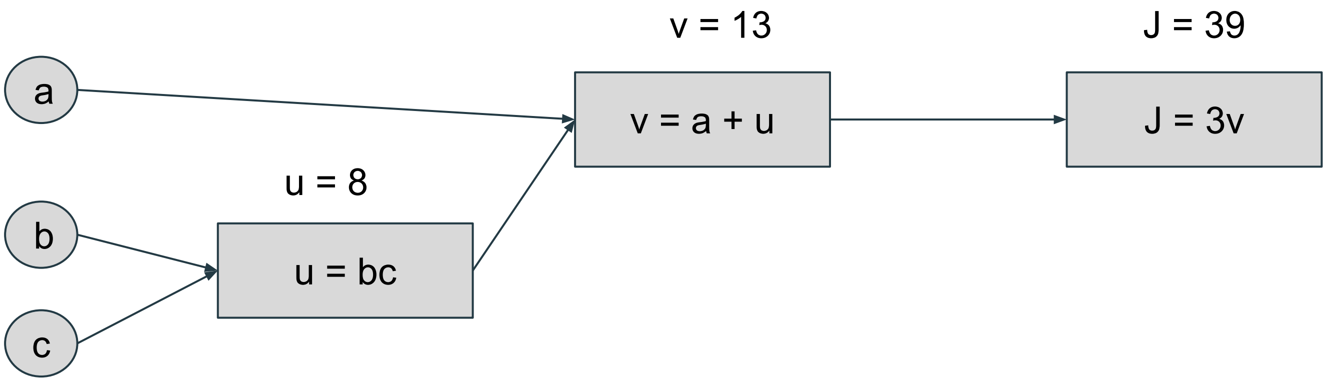
Showing an equation in graphical form.

Eg. J(a,b,c) = 3(a + bc)



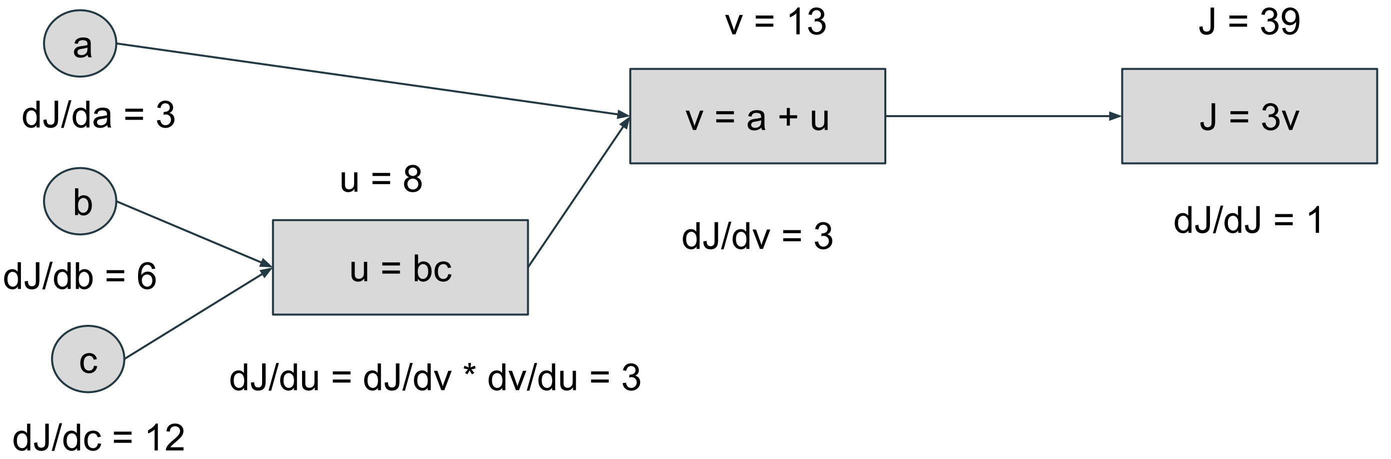
Forward Pass

If a = 5, b = 4, c = 2. Calculating the values of the nodes in topographical order is called forward pass



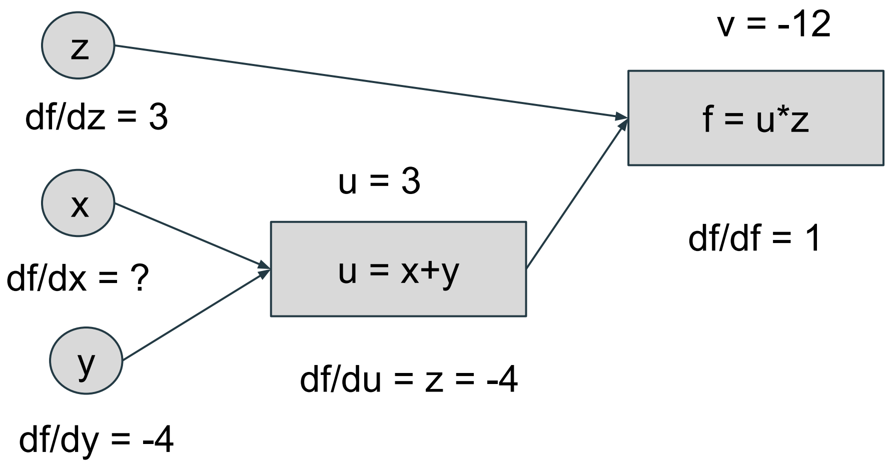
Calculating Gradient (Backward pass)

Now we want to calculate the gradient on the input with respect to the function, J = 3(a + bc)
So we will go from back to front, computing gradients of all the intermediate nodes in the graph.



Lets see more examples

f(x,y,z) = (x+y)z where x = -2, y = 5, z = -4



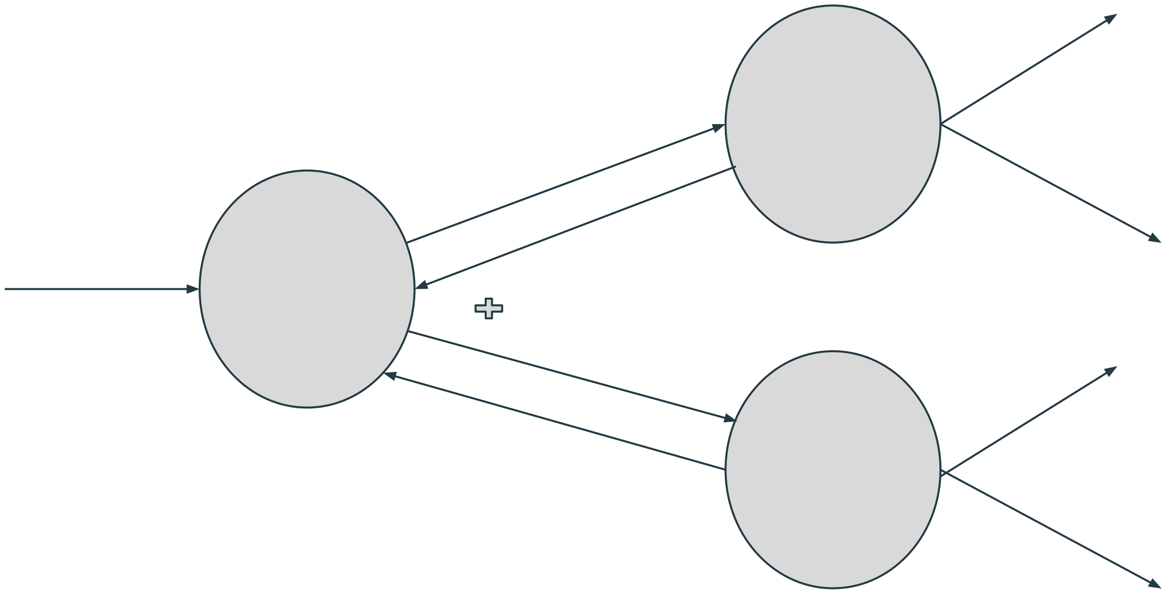
Interpretation of backpropagation

- df/dx means how much it influences the final output.
- So if df/dx > 0 (positive), Increasing x will increase the final output
- If df/dx < 0 (-ve), increasing x will decrease the final output

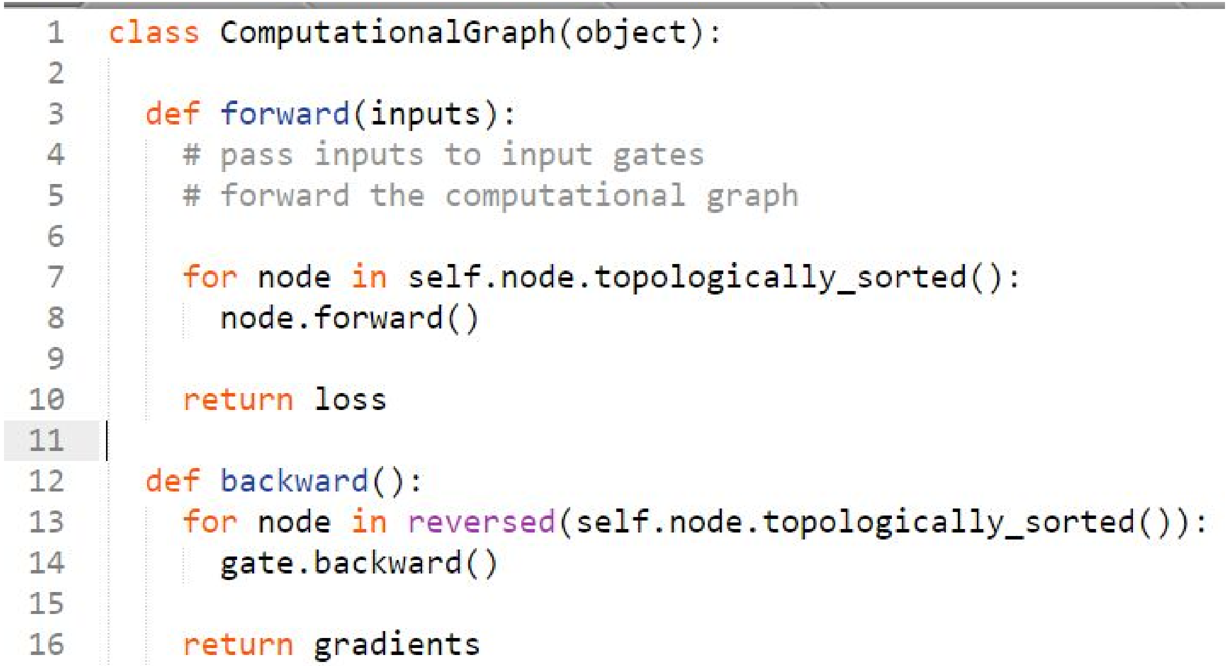
Patterns in backward flow

- Add gate is gradient distributor
- Max gate is gradient router
- Mul gate is kind of gradient switcher

Gradients add at branches

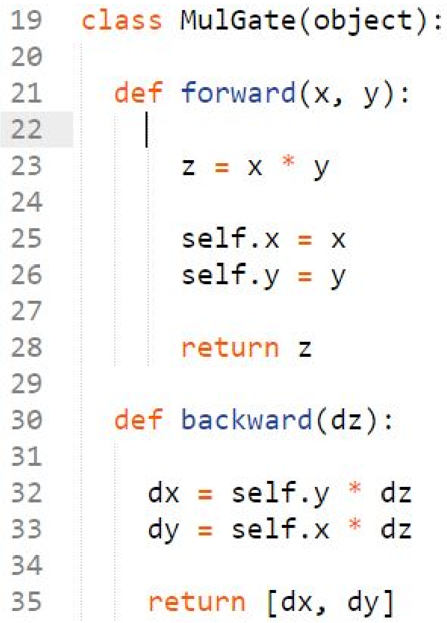


Implementation

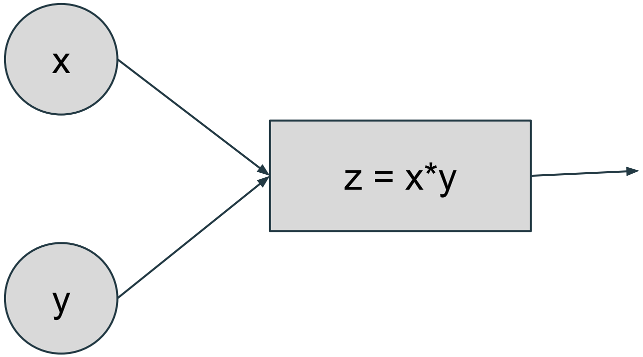


```
class ComputationalGraph(object):

  def forward(inputs):
    # pass inputs to input gates
    # forward the computational graph

    for node in self.node.topologically_sorted():
      node.forward()

    return loss

  def backward():
    for node in reversed(self.node.topologically_sorted()):
      gate.backward()

    return gradients
```

Implementation



```python
class MulGate(object):

  def forward(x, y):

      z = x * y

      self.x = x
      self.y = y

      return z

  def backward(dz):

    dx = self.y * dz
    dy = self.x * dz

    return [dx, dy]
```



Thanks