

# TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2022

## Backpropagation with Arrays and Tensors

## Program Values as Objects

Consider a scalar product ( $x$ ,  $y$  and  $z$  are each just real numbers).

$$z = xy$$

In a framework the values of the variables  $x$ ,  $y$   $z$  are objects in the sence of object oriented programming or Python.

In computing  $z.value$  we assume that  $x.value$  and  $y.value$  are known. In the base case these are are just inputs or parameters.

## Program Values as Objects

$$z = xy$$

The forward pass calls the procedure  $z.forward$  on each computed value  $z$ .

The object  $z$  holds its own inputs in its attributes.

Since  $z$  computed from a product the procedure  $z.forward$  assigns

$$z.value = x.value * y.value$$

## Backprop with Objects

$$z = xy$$

Each object  $x$  has an attribute  $x.\text{grad}$  which holds the gradient of the loss with respect to  $x$ .

We want

$$z.\text{grad} = \frac{\partial \mathcal{L}}{\partial z}$$

Backpropagation calls  $z.\text{backward}$  on each computed value  $z$  in the reverse order.

For  $z = xy$  we have that  $z.\text{backward}$  does

$$x.\text{grad} += y.\text{value} * z.\text{grad}$$

$$y.\text{grad} += x.\text{value} * z.\text{grad}$$

## Handling Arrays

Consider an inner product between vectors

$$z = x^\top y$$

In this case case `z.forward` does

$$z.value = 0$$

for  $i$  `z.value += x.value[i] * y.value[i]`

The backward procedure `z.backward` treats each `+=` instruction separately and does.

for  $i$  `x.grad[i] += y.value[i] * z.grad`

for  $i$  `y.grad[i] += x.value[i] * z.grad`

## Handling Arrays

Now consider multiplying a vector  $x$  by a matrix  $W$ .

$$y = Wx$$

In this case case `y.forward` does

```
for j  y.value[j] = 0
for i, j  y.value[j] += W.value[j, i] * x.value[i]
```

The backward procedure `y.backward` treats each individual `+=` as a scalar product and does

```
for i, j  x.grad[i] += W.value[j, i] * y.grad[j]
for i  W.grad[j, i] += x.value[i] * y.grad[j]
```

## A Linear Threshold Layer

$$s = \sigma (Wh - B)$$

for  $j$      $\tilde{s}[j] = 0$

for  $j, i$      $\tilde{s}[j] += W[j, i]h[i]$

for  $j$      $s[j] = \sigma(\tilde{s}[j] - B[j])$

Backpropagation is also done with loops treating each individual assignment and `+=` instruction.

## General Tensor Operations

In practice all deep learning source code can be written using scalar assignments and loops over scalar assignments. For example:

$$\text{for } h, i, j, k \quad Y[h, i, j] \quad += \quad A[h, i, k] \quad B[h, j, k]$$

has backpropagation loops

$$\begin{aligned} \text{for } h, i, j, k \quad A.\text{grad}[h, i, k] \quad += \quad Y.\text{grad}[h, i, j] \quad B.\text{value}[h, j, k] \\ \text{for } h, i, j, k \quad B.\text{grad}[h, j, k] \quad += \quad Y.\text{grad}[h, i, j] \quad A.\text{value}[h, i, k] \end{aligned}$$



**END**