TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2022

Backpropagation with Arrays and Tensors

Program Values as Objects

In a framework the program (or deep model) variables are objects in the sense of object oriented programming or Python.

Each object x stores its input objects in its instance variables and has an instance variable x.value storing its value.

The instance variable x.value is filled by sending x a forward message after its inputs have computed their values.

Each object x has an instance variable x.grad storing $\partial \mathcal{L}/\partial x$.

x.grad is filled by the backward methods of objects y that use x as an input. The backward method for y is called after y.grad has been filled and adds into x.grad for each input x.

Scalar Products

Consider a scalar product $z = xy$.

The forward method for z computes.

$$z.\mathrm{value} = x.\mathrm{value} * y.\mathrm{value}$$

The backward method for z computes

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

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

Handling Arrays

Consider an inner product between vectors

$$z = x^\top y$$

In this case case z.forward does

$$z.\mathrm{value} = 0$$

$$\mathrm{for}\ i\ \ z.\mathrm{value}\ \texttt{+=}\ x.\mathrm{value}[i] * y.\mathrm{value}[i]$$

The backward method for z treats each += instruction seperately and does.

$$\mathrm{for}\ i\ \ x.\mathrm{grad}[i]\ \texttt{+=}\ y.\mathrm{value}[i] * z.\mathrm{grad}$$

$$\mathrm{for}\ i\ \ y.\mathrm{grad}[i]\ \texttt{+=}\ x.\mathrm{value}[i] * z.\mathrm{grad}$$

Handling Arrays

Now consider multiplying a vector x by a matrix W.

$$y = Wx$$

In this case case y.forward does

$$\text{for } j \;\; y.\text{value}[j] = 0$$

$$\text{for } i, j \;\; y.\text{value}[j] \mathrel{\texttt{+=}} W.\text{value}[j, i] * x.\text{value}[i]$$

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

$$\text{for } i, j \;\; x.\text{grad}[i] \mathrel{\texttt{+=}} W.\text{value}[j, i] * y.\text{grad}[j]$$

$$\text{for } i, j \;\; W.\text{grad}[j, i] \mathrel{\texttt{+=}} x.\text{value}[i] * y.\text{grad}[j]$$

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:

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

has backpropagation loops

$$\text{for } h, i, j, k \;\; A.\text{grad}[h,i,k] \;\texttt{+=}\; Y.\text{grad}[h,i,j]\; B.\text{value}[h,j,k]$$
$$\text{for } h, i, j, k \;\; B.\text{grad}[h,j,k] \;\texttt{+=}\; Y.\text{grad}[h,i,j]\; A.\text{value}[h,i,k]$$

END