50.021 -AI

Alex

Week 03: Basics of neural networks

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

Suppose we have a layer which computes $Y = f(X)$ for some function f.

What needs to be done when programming such a layer?

- implement the forward pass. It takes X and computes $Y = f(X)$. Store variables from the forward pass needed for the backward pass.
- implement the backward pass. Two steps are necessary for this
  - if the layer has learnable parameters, such as the weights w and biases b in $Y = f(x) = x \cdot w + b$, then one need to compute the gradient $\frac{\partial E}{\partial w}$, $\frac{\partial E}{\partial b}$ - in order to be able to update w and b.

    The input to the backward pass is the gradient $\frac{\partial E}{\partial Y}$ with the respect to the layer outputs Y from the layer above. So one needs to compute (and similarly for a bias b)

    $$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial Y} \cdot \frac{\partial Y}{\partial w} \text{ by chain rule}$$
  - always: return the gradient $\frac{\partial E}{\partial X}$ with respect to this layer inputs, which are the outputs of a layer below.

    $$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y} \cdot \frac{\partial Y}{\partial X} \text{ by chain rule}$$

The example code uses DY in the backwardpass, this is $\frac{\partial E}{\partial Y}$ - where Y the outputs of the current layer. This is the gradient that comes from above

We need to do two things:

- if f has learnable parameters W, then: compute $\frac{dE}{dW} = DY * \frac{\partial f}{\partial W}$, store this as $self.DW$ – this needs NOT to be done for the tanh layer, but for $self.DB$, $self.DW$ in the linear layer

- always we have to compute

$$\frac{dE}{dX} = DY * \frac{\partial f}{\partial X}$$

and let the backward function return this term $\frac{dE}{dX}$

summing over batchsizes

One important trick: when running toolbox code, the toolboxes create of every layer as many copies as the batchsize (or they create a layer with dimensionality of all inputs and outputs being multiplied by the batch size). The forward computation runs in parallel for all samples in a batch.

When computing the gradient with respect to learnable variables W, one needs to sum up the gradients over all samples in the batch.

This comes from the formula of computing a SGD-minibatch update, where we sum over all samples in out minibatch:

$$\nabla_W \sum_{(x_i,y_i)\in Batch} E(f(x_i), y_i)$$

It says that we needs to be sum over all elements of the batch, when computing $\frac{dE}{dW}$ for updating W.

Example:

$$Y = X \cdot W$$

the input X for one sample would have shape $(1, K)$, the output for one sample would have shape $(1, M)$ (implies that W must have shape (K, M)!), and we have a batch size of 5. Then (up to transposition and reshaping) we know:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial Y} \cdot \frac{\partial Y}{\partial w} \text{ by chain rule}$$

and

$$X.shape = (5, K)\ W.shape = (K, M)$$
$$Y.shape = (5, M)$$
$$\frac{dE}{dY}.shape = Y.shape = (5, M)$$
$$\frac{dY}{dW}.shape = (5K, 1) \text{ or } (K, 5) \text{ or } (5, K) - \text{be aware of that additional 5}$$

You will need to sum over the additional dimension with the 5, when computing the chain rule

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial Y} \cdot \frac{\partial Y}{\partial w}$$