## **Computing derivatives w.r.t one weight in any layer**

1. Our roadmap for this module
   1. To calculate the desired gradient, we need to compute
   2. Gradient w.r.t output units
   3. Gradient w.r.t hidden units
   4. Gradient w.r.t weights and biases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Talk to the weight directly | Talk to the output layer | Talk to the previous hidden layer | Talk to the previous hidden layer | Talk to the weights |
|  |  | works for any number of output layers | |  |

* 1. For the rest of this exercise, our focus is on *Cross Entropy loss*  and *Softmax* output.

1. Here, what the sections highlighted in green are what we have covered so far, i.e. the derivative with respect to the last hidden layer and all other subsequent hidden layers
2. The gradients were calculated to be
3. Now, we will be computing the derivative of the loss function w.r.t weights and biases.
4. Recall that ak = bk + Wkhk-1
   1. k = layer number
   2. i = current layer neuron number
   3. j = previous/input layer neuron number
5. Now
6. We can use this to update the Weight by Gradient Descent
8. In the next step, we will look at updating all the weights in a layer simultaneously.