Last time: Gradients on computation graphs via backpropagation.

To implement this, each node (or layer) in our graph needs to define how to run forwards and backwards.

Let’s try and generalize the algorithm, starting from a concrete example.

We’ll consider binary classification with loss:

\[
\text{BCELoss} = -y \log \hat{y} - (1-y) \log (1-\hat{y})
\]
Let's try and generalize this.

Consider an arbitrarily deep network with
D feedforward layers (then an output layer).
So if we think about a simple fully connected layer, the local error is intuitive.
Exercise: Let’s think about implementation of an arbitrary layer such that it supports backprop.

```python
def forward(_):
    ...
def backward(_):
    ...
```

- What information do we need for backprop?

(See Exercise)
(Bonus Derivation)

1. \[
\frac{d}{dy} \left( \frac{g(y)}{y} \right) = \frac{\delta g}{\delta y} \frac{\delta y}{\delta y} = (g-y)
\]

\[
\frac{\delta}{\delta y} - y \ln y - (1-y) \ln (1-y)
\]

\[
= \frac{\delta}{\delta y} - y \ln y - y \frac{\delta}{\delta y} \ln (1-y)
\]

\[
= \left( \frac{\delta}{\delta y} - y \right) \cdot \ln y - y \frac{\delta}{\delta y} \ln (1-y)
\]

\[
= -y \frac{1}{y} = -\frac{y}{y}
\]

\[
= -\frac{y}{y} + \frac{(1-y)}{(1-y)}
\]

\[
\frac{\delta}{\delta y} - y \ln y - (1-y) \ln (1-y)
\]

\[
= (1-y) \frac{\delta}{\delta y} (1-y) \frac{1}{1-y}
\]

\[
= (1-y) \cdot 1 \cdot \frac{1}{1-y}
\]

\[
= \frac{(1-y)}{(1-y)}
\]

\[
\frac{\delta}{\delta y} = \frac{\delta \sigma}{\delta y} = \frac{\delta}{\delta y} \cdot (1-y)
\]

\[
\therefore \quad \frac{\delta}{\delta y} \cdot \frac{\delta}{\delta y} = \left( -\frac{y}{y} + \frac{(1-y)}{(1-y)} \right) \cdot \frac{\delta}{\delta y} (1-y)
\]

\[
= \left( \frac{y(1-y)}{(1-y)} \right) \cdot \frac{y}{y} + (\frac{1}{1-y}) \cdot \frac{(1-y)}{(1-y)}
\]

\[
= \left( \frac{y}{y} \right) + \frac{(1-y)}{(1-y)}
\]

\[
= 1 + 1
\]

\[
= 2
\]
= (1-\hat{y}) \cdot \hat{y} + \hat{g}(1-\hat{y})
= -y(1-\hat{y}) + \hat{g}(1-\hat{y}) = -y + \hat{g} \hat{y} + \hat{g} - \hat{g} \hat{y} = \hat{y} - y