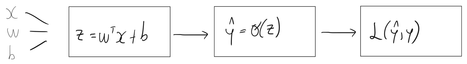
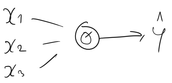
- Logistic regression primer




- Loss function

$$L(\hat{y}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$

$$\frac{\partial L_{sc}}{\partial \hat{y}}\left[-y\log(\hat{y}) - (1-y)\log(1-\hat{y})\right] \Rightarrow -\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}$$

- Derivatives

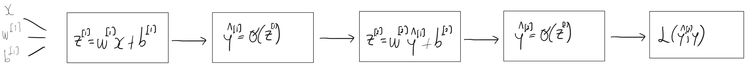
$$\frac{\partial \hat{y}}{\partial z} = \frac{e^z}{(1+e^z)^2} = \hat{y}(1-\hat{y})$$

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \left(-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}\right)\left(\hat{y}(1-\hat{y})\right) \Rightarrow \hat{y} - y$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial w} = (\hat{y} - y)x$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial b} = \hat{y} - y$$

# • Neural network




- Derivatives

$$\frac{\partial L}{\partial \hat{y}^{[2]}} = -\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}$$

$$\frac{\partial L}{\partial z^{[2]}} = \frac{\partial L}{\partial \hat{y}^{[2]}} \frac{\partial \hat{y}^{[2]}}{\partial z^{[2]}} = \hat{y}^{[2]} - y$$

$$\frac{\partial L}{\partial W^{[2]}} = \frac{\partial L}{\partial \hat{y}^{[2]}} \frac{\partial \hat{y}^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial W^{[2]}} = (\hat{y}^{[2]} - y)\hat{y}^{[1]T}$$

$$\frac{\partial L}{\partial b^{[2]}} = \frac{\partial L}{\partial \hat{y}^{[2]}} \frac{\partial \hat{y}^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial b^{[2]}} = \hat{y}^{[2]} - y$$

$$\frac{\partial L}{\partial \hat{y}^{[1]}} = \frac{\partial L}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial \hat{y}^{[1]}} = W^{[2]T}(\hat{y}^{[2]} - y)$$

$$\frac{\partial L}{\partial z^{[1]}} = \frac{\partial L}{\partial \hat{y}^{[1]}} \frac{\partial \hat{y}^{[1]}}{\partial z^{[1]}} = W^{[2]T}(\hat{y}^{[2]} - y) * \sigma'(z^{[1]})$$

$$\frac{\partial L}{\partial W^{[1]}} = \frac{\partial L}{\partial z^{[1]}} \frac{\partial z^{[1]}}{\partial W^{[1]}} = (\hat{y}^{[2]} - y)W^{[2]T} x^T$$

$$\frac{\partial L}{\partial b^{[1]}} = \frac{\partial L}{\partial z^{[1]}} \frac{\partial z^{[1]}}{\partial b^{[1]}} = \frac{\partial L}{\partial z^{[1]}} x^T$$

$$\frac{\partial L}{\partial b^{[1]}} = \frac{\partial L}{\partial z^{[1]}} \frac{\partial z^{[1]}}{\partial b^{[1]}} = \frac{\partial L}{\partial z^{[1]}}$$

- Key equations in a vectorized form

$$dz^{[2]} = \hat{y}^{[2]} - y \qquad dz^{[2]} = (\hat{y}^{[2]} - y)W^{[2]T} * \sigma'(z)$$

$$dW^{[2]} = \frac{1}{m}\left[dz^{[2]} \cdot y\right]\hat{y}^{[1]T} \qquad dW^{[2]} = \frac{1}{m} dz^{[2]} x^T$$

$$db^{[2]} = \frac{1}{m}\sum\left[dz^{[2]} \cdot y\right] \qquad db^{[2]} = \frac{1}{m}\sum dz^{[2]}$$

- Updating the weights and biases

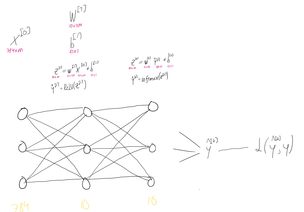
$$W^{[2]} = W^{[2]} - \alpha\, dW^{[2]} \qquad W^{[2]} = W^{[2]} - \alpha\, dW^{[2]}$$

$$b^{[2]} = b^{[2]} - \alpha\, db^{[2]} \qquad b^{[2]} = b^{[2]} - \alpha\, db^{[2]}$$

First layer — Second layer

$\alpha$ = learning rate

- Dimensions of a neural network




- Flow

1. Feed some data, forward propagation
2. Calculate error
3. Change $W$'s and $b$'s based on the error, backpropagation

- Chain rule

If $y = f(u)$, where $u = g(x)$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

- Backpropagation

The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule; this is an example of dynamic programming.