Section 08: Neural Networks

1. The Chain Rule

(a) Let $f\colon \mathbb{R}^n \to \mathbb{R}^m$, $g\colon \mathbb{R}^\ell \to \mathbb{R}^n$. Write the Jacobian of $f \circ g$ as a matrix in terms of the Jacobian matrix $\frac{\partial f}{\partial y}$ of f and the Jacobian matrix $\frac{\partial g}{\partial x}$ of g. Make sure the matrix dimensions line up. What conditions must hold in order for this formula to make sense?

(b) Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}^n$. Write the derivative of $f \circ g$ as a summation between the partial derivatives $\frac{\partial f}{\partial y_i}$ of f and the partial derivatives $\frac{\partial g_i}{\partial x}$ of g.

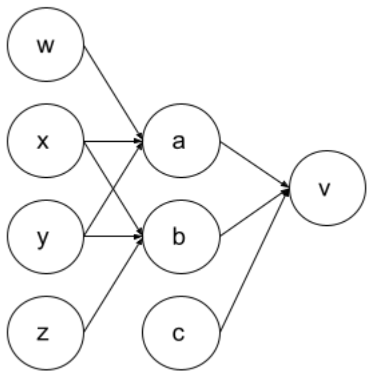
(c) What if instead the input of g is a matrix $W \in \mathbb{R}^{p \times q}$? Can we still represent the derivative $\frac{\partial g}{\partial W}$ of g as a matrix?

2. Neural Network Chain Rule Warm-Up

Consider the following equations:

$$v(a, b, c) = c(a - b)^2$$
$$a(w, x, y) = (w + x + y)^2$$
$$b(x, y, z) = (x - y - z)^2$$

The way variables are related to each other can be represented as the network:



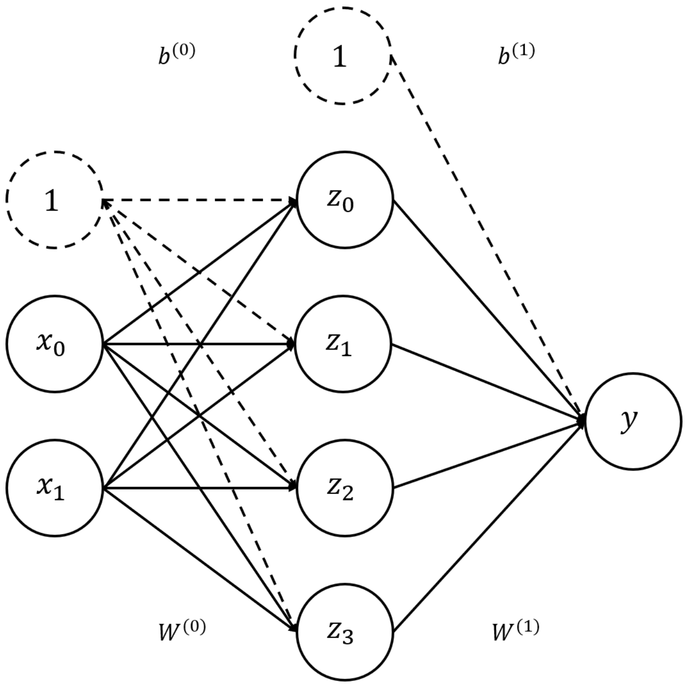
(a) Using the multi-variate chain rule(part 1.b), write the derivatives of the output v with respect to each of the input variables: c, w, x, y, z using only partial derivative symbols.

(b) Compute the values of all the partial derivatives on the RHS of your results to the previous question. Then use them to compute the values on the LHS.

3. 1-Hidden-Layer Neural Network Gradients and Initialization

3.1. Forward and Backward pass

Consider a 1-hidden-layer neural network with a single output unit. Formally the network can be defined by the parameters $W^{(0)} \in \mathbb{R}^{h \times d}$, $b^{(0)} \in \mathbb{R}^h$; $W^{(1)} \in \mathbb{R}^{1 \times h}$ and $b^{(1)} \in \mathbb{R}$. The input is given by $x \in \mathbb{R}^d$. We will use sigmoid activation for the first hidden layer z and no activation for the output y. Below is a visualization of such a neural network with $d = 2$ and $h = 4$.



(a) Write out the forward pass for the network using $x, W^{(0)}, b^{(0)}, z, W^{(1)}, b^{(1)}, \sigma$ and y.

Hint: Write $z = \ldots$ and $y = \ldots$

(b) Find the partial derivatives of the output with respect $W^{(1)}$ and $b^{(1)}$, namely $\frac{\partial y}{\partial W^{(1)}}$ and $\frac{\partial y}{\partial b^{(1)}}$.

(c) Now find the partial derivative of the output with respect to the output of the hidden layer z, that is $\frac{\partial y}{\partial z}$

(d) Finally find the partial derivatives of the output with respect to $W^{(0)}$ and $b^{(0)}$, that is $\frac{\partial y}{\partial W^{(0)}}$ and $\frac{\partial y}{\partial b^{(0)}}$.

Hint: First find $\frac{\partial z_i}{\partial W_i^{(0)}}$ and $\frac{\partial z_i}{\partial b_i^{(0)}}$, where $W_i^{(0)}$ denotes the i-th row of $W^{(0)}$. Then note that $\frac{\partial y}{\partial W_i^{(0)}} = \sum_{j=1}^{h} \frac{\partial y}{\partial z_j} \frac{\partial z_j}{\partial W_i^{(0)}} = \frac{\partial y}{\partial z_i} \frac{\partial z_i}{\partial W_i^{(0)}}$ and $\frac{\partial y}{\partial b_i^{(0)}} = \sum_{j=1}^{h} \frac{\partial y}{\partial z_j} \frac{\partial z_j}{\partial b_i^{(0)}} = \frac{\partial y}{\partial z_i} \frac{\partial z_i}{\partial b_i^{(0)}}$ using the chain rule for multi-variate functions(1.b).

3.2. Weight initialization

Suppose we initialize all weights and biases in the network to 0 before performing gradient descent.

(a) For all $x \in \mathbb{R}^d$, find z and y after the forward pass.

(b) Now find the values of the gradients $\frac{\partial y}{\partial W^{(1)}}$, $\frac{\partial y}{\partial b^{(1)}}$, $\frac{\partial y}{\partial W^{(0)}}$ and $\frac{\partial y}{\partial b^{(0)}}$. Note that some of the gradients will be in terms of x.

(c) Observe the values of each z_i and observe each $\frac{\partial y}{\partial W_i^{(l)}}$ and $\frac{\partial y}{\partial b_i^{(l)}}$. What do you notice? And what does this imply for the expressiveness of the network? (Note that there is nothing special about the value 0 here, it just simplifies the calculations. The same can be shown for initialization with any constant c)