Q5

Wednesday, February 22, 2023 1:11 PM

$x = [1, x1, x2]$

$X = np.array([[1, x_1, x2]])$

$w1 = np.array([[w1, w2], [w3, w4], [w5, w6]])$ $\begin{bmatrix} w1, w2 \\ w3, w4 \\ w5, w6 \end{bmatrix}$

$w2 = np.array([[v_1], [v_2], [v_3]])$ $\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$

If we denote $g(X.w1) = np.array([[1, z_1, z_2]])$, then

$g(g(X.w1).w2)$ can be written: $g(v1 \cdot 1 + v2 \cdot z1 + v3 \cdot z2)$

Let $g(x)$ be a linear activation fn s.t. $g(x) = x$

a. $g(v1 \cdot 1 + v2 \cdot z1 + v3 \cdot z2)$

$= g(g(x.w1).w2)$ $x.w_1 = [w_1, w_2] + [w3x1, w4x1] + [w_5, w_6]x2$

$X.w1 = [w1 + x1w3 + x2w5, w2 + x1w4 + x2w6]$

$\therefore g(v1 \cdot 1 + v2 \cdot z1 + v3 \cdot z2) = g(g([w1 + x1w3 + x2w5, w2 + x1w4 + x2w6]).w2)$

b. Assuming $g(x) = x$, we can simplify our expression from above:

Start with: $g(g([w1 + x1w3 + x2w5, w2 + x1w4 + x2w6]).w2)$

$\rightarrow g(([w_1, w_2]1 + [w3, w4]x1 + [w_5, w_6]x2).w2)$

$\rightarrow g(v1[w_1, w_2] \cdot 1 + v2[w_3, w_4] \cdot x_1 + v3[w_5, w_6] \cdot x_2)$

Clearly we have written $g(v1 \cdot 1 + v2 \cdot z1 + v3 \cdot z2)$ as

$g(m \cdot 1 + n \cdot x1 + o \cdot x2)$ where $m = v1[w_1, w_2]$,

$n = v2[w3, w4]$, $o = v3[w5, w6]$

c. Since the result of this MLP using linear activation functions can be written in terms of the original input (only with modified weights), our MLP isn't able to learn anything more complicated than a single perceptron (which would do the exact same thing of assigning weights to each of the input features).

d. Now let $g(x) = (1 + exp(-x))^{-1}$

Start with: $g(g([w1 + x1w3 + x2w5, w2 + x1w4 + x2w6]).w2)$

$\rightarrow g(\frac{1}{\ldots\ 2 + x1w4 + x2w6])}.w2)$

$$\rightarrow g\left(\frac{1}{1+\exp\left(-[w1+x1w3+x2w5,\ w2+x1w4+x2w6]\right)}\cdot w2\right)$$

Clearly the exponential term cannot be written as a linear combination of 1, x1, and x2 (due to the sigmoid function being non linear itself), so we cannot write its dot product with w2 as a linear combination of 1, x1, and x2 either.

e. Since a combination of sigmoid perceptrons cannot be reduced to a single sigmoid perceptron with altered weights, a MLP using sigmoid activation functions can learn more complicated decision boundaries than a single sigmoid perceptron.

f. An activation must be nonlinear and differentiable in order to be useful in a neural network. Nonlinearity allows the network to learn arbitrarily complicated decision boundaries, while differentiability allows backpropagation through the network to work properly by allowing the use of gradient descent to update the weights of each node.