Formal proof for backpropogation with ReLU

$$\frac{\partial a}{\partial w} = \overset{④}{\frac{\partial a}{\partial z}} \overset{③}{\frac{\partial z}{\partial s}} \overset{②}{\frac{\partial s}{\partial H}} \overset{①}{\frac{\partial H}{\partial w}}$$

$$s = \text{sum}(\underbrace{w \odot x}_{H})$$

By Jacobian differentiation,

① $$\frac{\partial H}{\partial w} = \begin{array}{c} n \\ \begin{bmatrix} x_1 & 0 & \cdots & 0 & 0 \\ 0 & x_2 & & & \vdots \\ \vdots & & x_3 & & \vdots \\ \vdots & & & \ddots & \\ 0 & \cdots & & 0 & x_n \end{bmatrix} \end{array}$$ n inputs

② By Jacobian & derivative of a sum

$$\sum_{i=1}^{n} \overset{H_1, H_2 \ldots H_n}{w_i x_i} = [\,-\, s \,-\,]$$

$$\frac{\partial s}{\partial H} = [\underbrace{1 \; 1 \; 1 \; 1 \; \ldots \; 1}_{n \text{ times}}]$$ Each derivative has only 1 component that returns a 1.

③ $z = s - b$

$$\frac{\partial z}{\partial s} = 1 \quad (\text{Scalar})$$

④ 0 or 1, depending on if $z \geq 0$, or $z \leq 0$ (Scalar)

End result:

$$\frac{\partial a}{\partial w} = (1)\,1\,[\;\;]^{n}\left[\;\;\right]^{n}$$

$$= 1\,[x_1, x_2\, x_3, \ldots, x_n]^{n}$$

For $\frac{\partial a}{\partial b}$, $\frac{\partial a}{\partial b} = \frac{\partial a}{\partial z} \cancel{\frac{\partial z}{\partial b}}\, 1$

$$= 0 \text{ or } 1 \quad (\text{Scalar})$$

Scanned with CamScanner

desired outputted

$$MSE: \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial a}\frac{\partial a}{\partial w}$$

let $v = y - a^L$

$$\frac{1}{2m}\sum_{i=1}^{m}(v)^2$$

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial v}\frac{\partial v}{\partial w}$$

(I) (II)

$$\frac{\partial v}{\partial w} = \frac{\partial}{\partial w}(y - a^L)$$ where $v = y - a^L$

$$\frac{\partial v}{\partial w} = -(1)\left(\frac{\partial a}{\partial w}\right)$$ (chain rule again)

(I) $$\frac{\partial C}{\partial v} = \frac{1}{2m}\sum_{i=1}^{m}(2v)$$

$$= \frac{1}{m}\sum_{i=1}^{m} v$$

$$\frac{\partial C}{\partial v} = \frac{1}{m}\sum_{i=1}^{m}(v)$$

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial v}\frac{\partial v}{\partial w}$$

$$= \frac{1}{m}\sum_{i=1}^{m} v(-1)\left(\frac{\partial a}{\partial w}\right)$$

$$= \frac{1}{m}\sum_{i=1}^{m}\begin{cases} -(0)^T \\ -v\frac{\partial z}{\partial w}\end{cases}$$

$$\frac{\partial C}{\partial w} = \frac{1}{m}\sum_{i=1}^{m}\begin{cases} 0^T \\ -(y_i - (w^T x + b))(x_1\ x_2\ \dots\ x_n)\end{cases}$$

if $w^T x + b \geq 0$, there is no need for a second max function.

$$\frac{\partial C}{\partial w} = \frac{1}{m}\sum_{i=1}^{m}\underbrace{(w^T x + b - y_i)}_{\text{Scalar term}}[x_1, x_2\ \dots\ x_n]$$

what does this looks like?

if we define $(w^t x + b - y_i)$ as e_i

$$\frac{1}{m}\begin{bmatrix} e_1 x_1 + e_2 x_2 + \dots + e_n x_1 \\ e_2 x_2 + e_2 x_2 + \dots + e_n x_2 \\ \vdots \\ e_n x_n \; e_2 x_n + \dots + e_m x_n \end{bmatrix}$$

$\frac{\partial c}{\partial w}$ wrt all ws, averaged over all training examples.

Derivative of the cost wrt the bias:

$$\frac{\partial c}{\partial b} = \frac{\partial v}{\partial a}\frac{\partial a}{\partial b}$$

$$= (-1)(v)\begin{cases} 0 \\ 1 \end{cases}$$

$$= \begin{cases} 0 \\ -(y - a^i) \end{cases}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\underbrace{(w^t x + b - y)}_{e_i} = \frac{1}{m}\sum_{i=1}^{m} e_i \quad (\text{Scalar})$$

Four equations of back propogation :

$$\delta_j^l = \frac{\partial C}{\partial a_j^l} \sigma'(z_j^l)$$

① $\delta^L = \nabla_a C \odot \sigma'(z^L)$

② $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$

③ $\frac{\partial C}{\partial b} = \delta$

④ $\frac{\partial C}{\partial w} = (a^{l-1})^T \delta^l$

Imagine we want our neural network to be 784×30×10

self.num_layers = len(sizes) = 3

passing in self.sizes = [784, 30, 10]

Looping through sizes 30, 10

Need a bias vector of 30×1, and 10×1

Biases.append(np.random.randn(y,1)) => creates a list of numpy arrays

(30×1), (10×1) numpy arrays

weights need to be 30×784, 10×30 for dot products

using the zip function,

zip(sizes[:-1], sizes[1:]) -> x from first, up excluding last

-> y from second, excluding last

x: 784, 30,

y: 30,10

(y,x) = 30×784, 10×30

starting from the input, feedforward runs through a loop

For a training example 784×1

$$784\overset{1}{\begin{bmatrix} \\ \\ \end{bmatrix}} \overset{784}{\begin{bmatrix} & & \\ & & \end{bmatrix}}30 + \overset{1}{\begin{bmatrix} \\ \\ \end{bmatrix}}30$$

, then apply sigmoid element wise to all elements

$$30\overset{1}{\begin{bmatrix} \\ \\ \end{bmatrix}} \overset{30}{\begin{bmatrix} & & \\ & & \end{bmatrix}}10 + \overset{1}{\begin{bmatrix} \\ \end{bmatrix}}10$$

, apply sigmoid element wise

length of test data => n-test = 10,000

n = length of training data = 50,000

Looping through epochs,

Mini-batches of size mini_batch_size => # of columns = batch size

# of epochs x # of batches = # of times gradient descent is performed. => batches = 500, epochs = 10 => 5000 times

Scanned with CamScanner

Create weight and bias matrices, matching the shapes already initialized

We first initialize the matching matrices with zeros. Running back propagation, we set the list of matrices that match the shapes of the gradient matrices.

Then, we subtract the gradients, after multiplying by eta / # of training examples

In the back propagation algorithm, we are passing in a batch of training examples.

Initialize the activation x, and activations as a list. Create a list of the weighted sums, and activations

① $\delta^L = \nabla_a C \odot \sigma'(z^L)$

② $\delta^l = ((w^{l+1})^T \delta^{l+1} \odot \sigma'(z^l))$

③ $\frac{\partial C}{\partial b} = \delta$

④ $\frac{\partial C}{\partial w} (a^{l-1})^T \delta^l$

From neural networks to implementation of back propogation:

eg. 2 inputs of 784 pixels

$$a_1 = {}_{15,\,j}\left[\;\right]^{784\ k} \left[\;\right]_{784}^{2} + \left[\;\right]_{15}^{2} \Rightarrow \left[\;\right]_{15}^{2}$$

15th neuron, into the jth neuron

$$_{10}\left[\;\right]^{15} \left[\;\right]_{15}^{2} + \left[\;\right]_{10}^{2} = \left[\;\right]_{10}^{2} = a_2$$

$$_{784}\left(\;\right)^{100} \left(\;\right)_{10}^{784} + \left(\;\right)_{10}^{100}$$

$$\overset{w}{\left(\;\right)^{10}}\ \cancel{100}\ \overset{x}{\left(\;\right)^{100}} = {}_{10}\left(\;\right)^{100}\ \ {}_{10}\left(\;\right)^{100}$$