

Backpropagation: Calculus

Backpropagation calculus _ Chapter 4, Deep learning.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$

Chain rule

$$C_0(\dots) = (a^{(L)} - y)^2$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma(z^{(L)})$$

Desired output

03:55 10:17

This flowchart shows how the change in weight affect the change in $z^{(L)}$ which affects the change in $a^{(L)}$ which finally affects the change in cost C_0

σ is the sigmoid activation function, which is not used much in the present times, rather *Tanh* or *ReLU* is used due to slow convergence of Sigmoid and 0 centric base in Tanh and finally, fast computation of ReLU.

Backpropagation calculus _ Chapter 4, Deep learning.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$

$$\frac{\partial C_0}{\partial a^{(L)}} = 2(a^{(L)} - y)$$

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'(z^{(L)})$$

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}$$

$$C_0 = (a^{(L)} - y)^2$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma(z^{(L)})$$

05:00 10:17

Backpropagation calculus _ Chapter 4, Deep learning.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} = a^{(L-1)} \sigma'(z^{(L)}) 2(a^{(L)} - y)$$

Average of all training examples

$$\underbrace{\frac{\partial C}{\partial w^{(L)}}}_{\text{Derivative of full cost function}} = \frac{1}{n} \sum_{k=0}^{n-1} \frac{\partial C_k}{\partial w^{(L)}}$$

$$C_0 = (a^{(L)} - y)^2$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma(z^{(L)})$$

05:24 10:17

This expression is just one component of the gradient vector which is built up from the partial derivatives of all those weights and biases.

Backpropagation calculus _ Chapter 4, Deep learning.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} = a^{(L-1)} \sigma'(z^{(L)}) 2(a^{(L)} - y)$$

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \vdots \\ \frac{\partial C}{\partial w^{(L)}} \\ \frac{\partial C}{\partial b^{(L)}} \end{bmatrix}$$

$$C_0 = (a^{(L)} - y)^2$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma(z^{(L)})$$

05:37 10:17

sensitivity w.r.t. **bias** is almost identical by replacing the $\partial w^{(L)}$ by $b^{(L)}$ whose derivative is = 1.

So, the final expression for sensitivity w.r.t. bias becomes:

$$\frac{\partial C_0}{\partial w^{(L)}} = \sigma'(z^{(L)}) 2(a^{(L)} - y)$$

NOTE

The calculations above are for a model which has a single neuron in all of its layers, though the above work is quite similar for a more complex neural network structure.

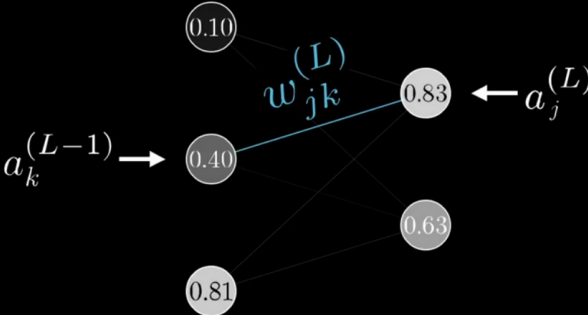
Calculations for more complex neural network

Notations:

Backpropagation calculus _ Chapter 4, Deep learning.mp4 - VLC media player

$$z_j^{(L)} = w_{j0}^{(L)} a_0^{(L-1)} + w_{j1}^{(L)} a_1^{(L-1)} + w_{j2}^{(L)} a_2^{(L-1)} + b_j^{(L)}$$

$$a_j^{(L)} = \sigma(z_j^{(L)})$$

$$C_0 = \sum_{j=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$


08:04 10:17

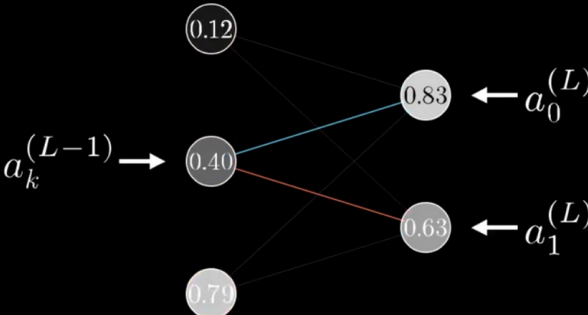
Final Equations:

Backpropagation calculus _ Chapter 4, Deep learning.mp4 - VLC media player

$$\frac{\partial C_0}{\partial a_k^{(L-1)}} = \underbrace{\sum_{j=0}^{n_L-1} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}}}_{\text{Sum over layer L}}$$

$$z_j^{(L)} = \dots + w_{jk}^{(L)} a_k^{(L-1)} + \dots$$

$$a_j^{(L)} = \sigma(z_j^{(L)})$$

$$C_0 = \sum_{j=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$


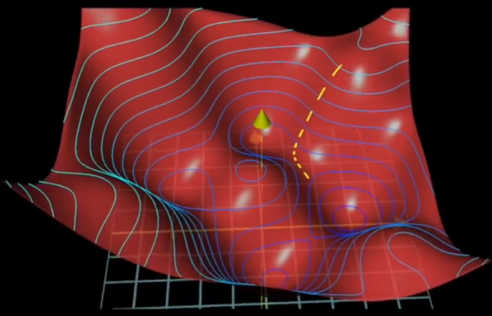
09:06 10:17

Notice the slight change... instead of taking derivative of cost w.r.t. weight, we took it w.r.t. the k th activation in the layer $L-1$

n_L represents the number of neurons in the layer L .

Backpropagation calculus _ Chapter 4, Deep learning.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help



$$\nabla C \leftarrow \left\{ \begin{array}{l} \frac{\partial C}{\partial w_{jk}^{(l)}} = a_k^{(l-1)} \sigma'(z_j^{(l)}) \frac{\partial C}{\partial a_j^{(l)}} \\ \sum_{j=0}^{n_{l+1}-1} w_{jk}^{(l+1)} \sigma'(z_j^{(l+1)}) \frac{\partial C}{\partial a_j^{(l+1)}} \\ \text{or} \\ 2(a_j^{(L)} - y_j) \end{array} \right.$$

09:37 10:17 125%