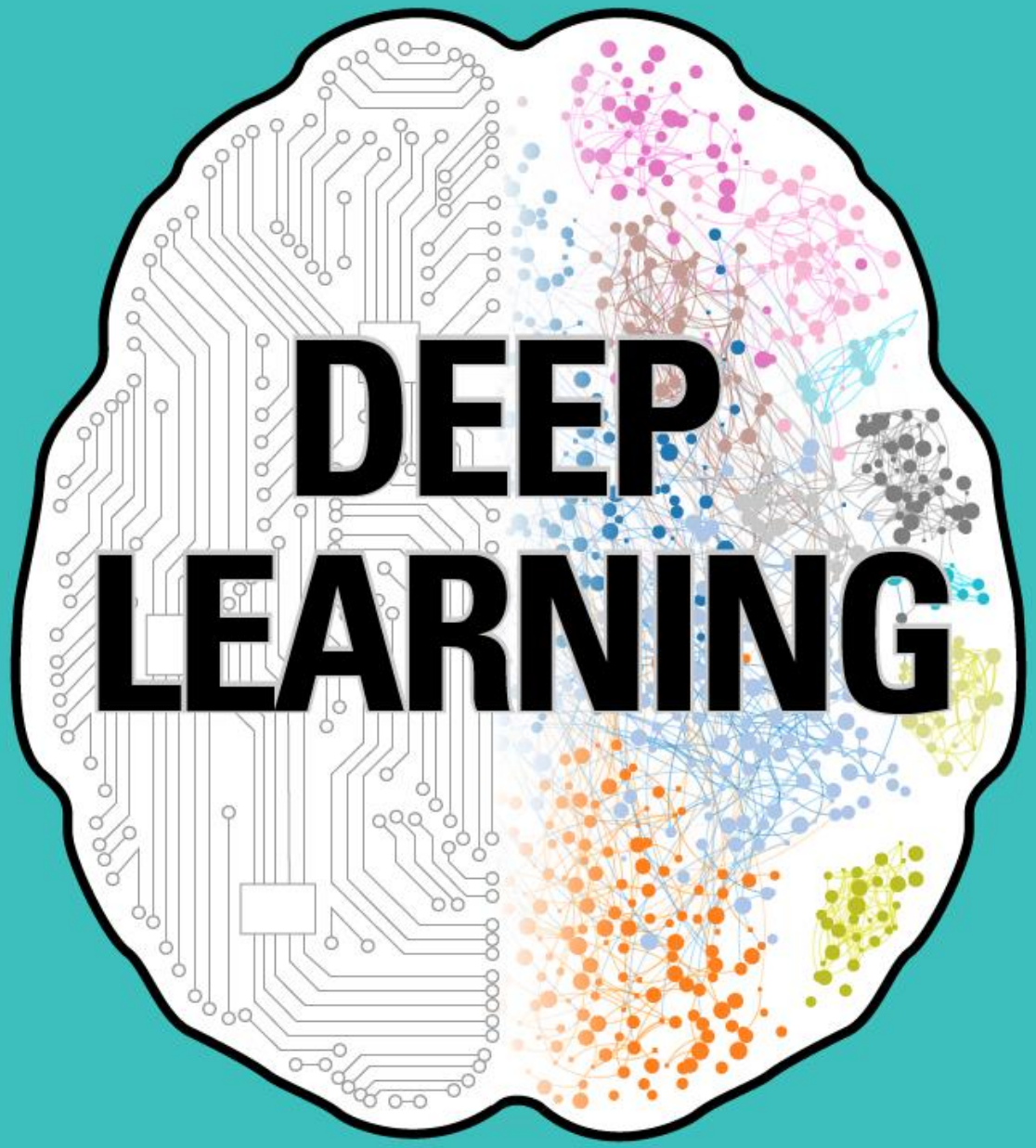


Neural Networks & Deep Learning

http://neuralnetworksanddeeplearning.com/chap5.html

The vanishing gradient problem

Speed of learning

we'll find that there's an intrinsic instability associated to learning by gradient descent in deep, many-layer neural networks. This instability tends to result in either the early or the later layers getting stuck during training.

1. Error at the output layer

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

2. Error relationship between two adjacent layers

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

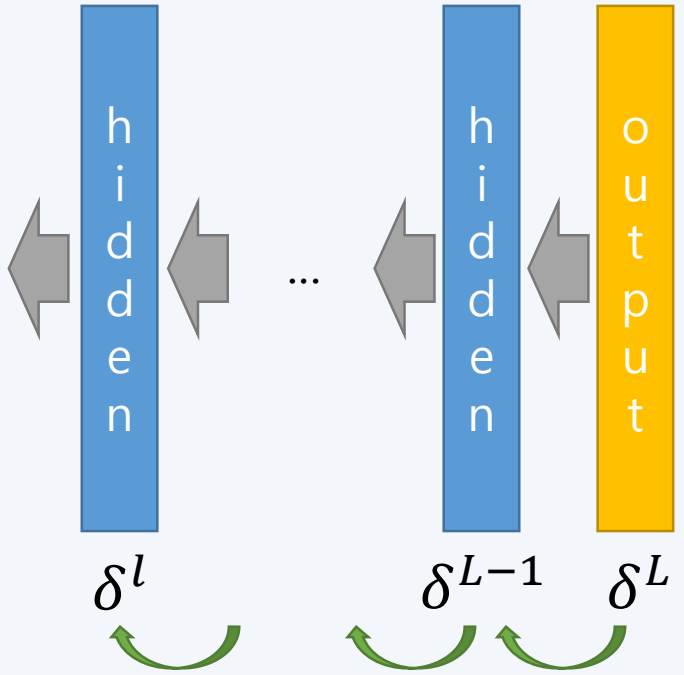
3. Gradient of C in terms of bias

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

4. Gradient of C in terms of weight

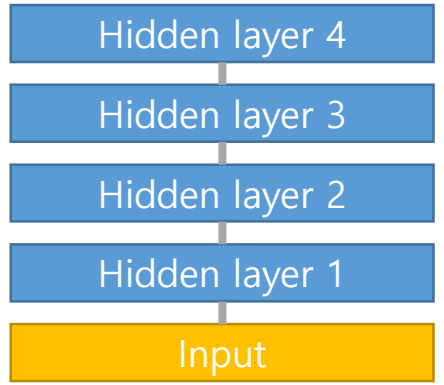
$$\frac{\partial C}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1}$$

GD+BP



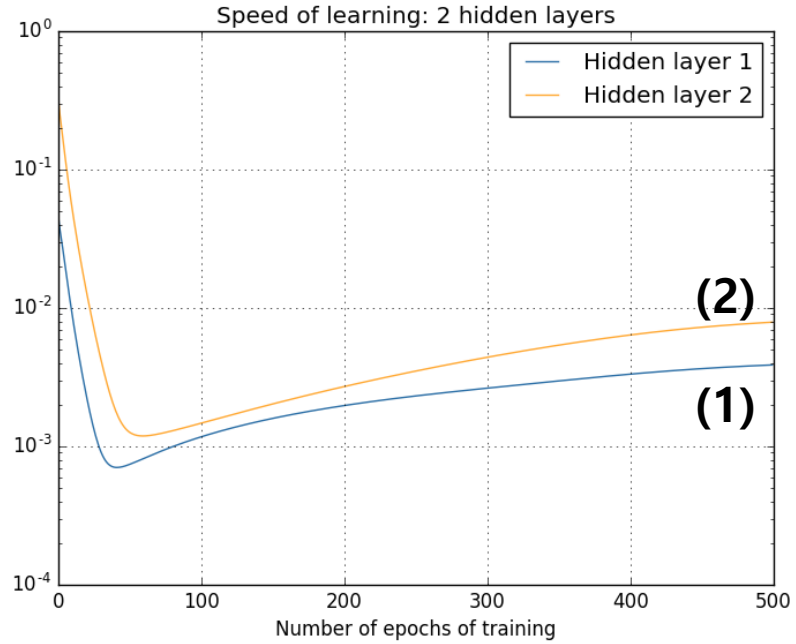
$\|\delta^l\|$ measures the speed at which the hidden layer is learning

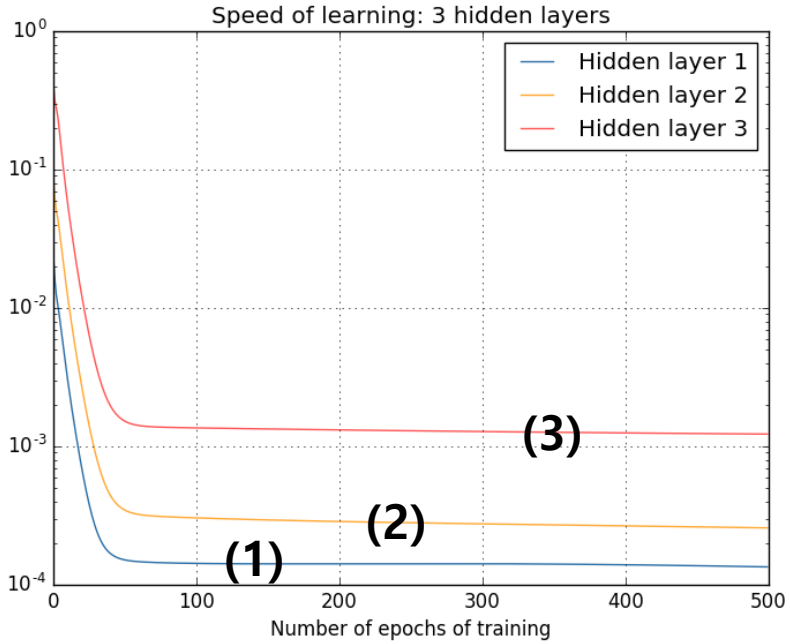
The vanishing gradient problem

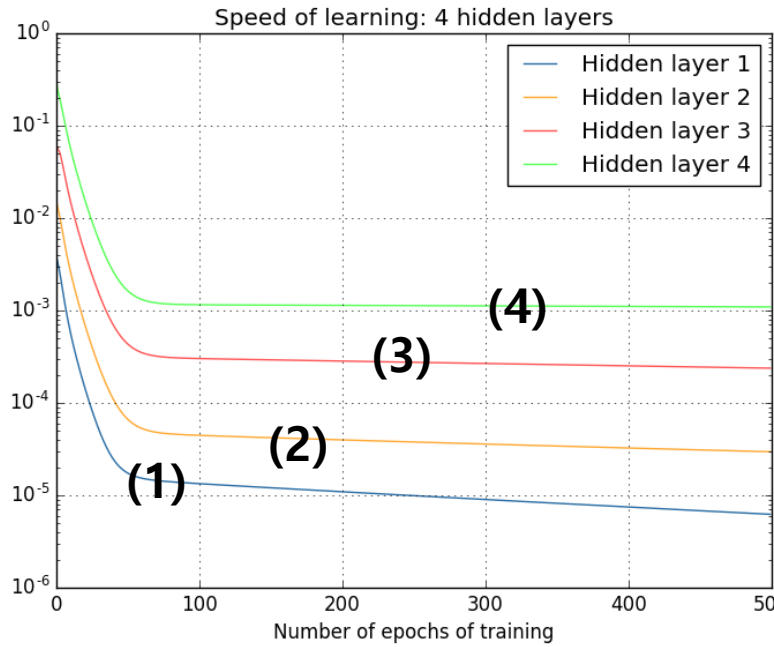


An important observation

- All hidden layers has 30 neurons
- No mini-batches (mini-batch gives more noisy data)
- 10000 training images



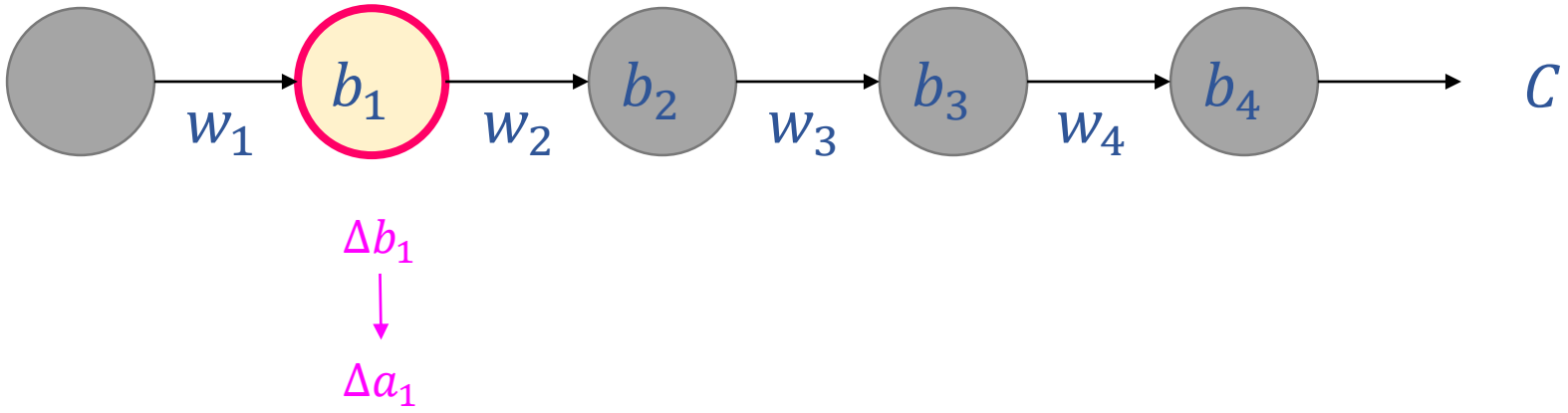




- **In at least some deep neural networks, the gradient tends to get smaller as we move backward through the hidden layers. (vanishing gradient)**
- **Opposite case is called exploding gradient**

What's causing the vanishing gradient problem?

A simple network with 4 hidden layers



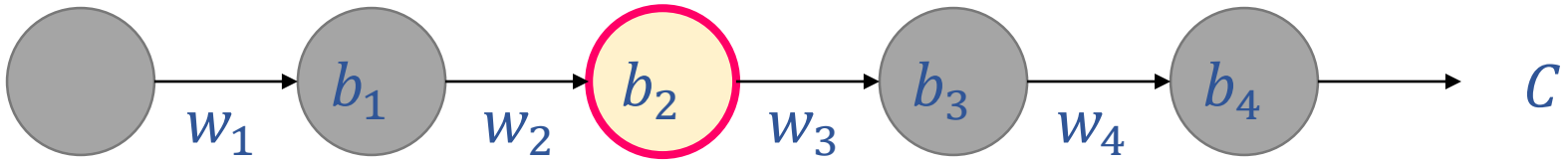
$$\frac{\partial a_1}{\partial b_1} = \frac{\partial}{\partial b_1}\sigma(w_1 a_0 + b_1) = \sigma'(w_1 a_0 + b_1) = \sigma'(z_1)$$

$$\frac{\Delta a_1}{\Delta b_1} \approx \frac{\partial a_1}{\partial b_1} = \sigma'(z_1)$$

$$\Delta a_1 = \sigma'(z_1)\Delta b_1$$

What's causing the vanishing gradient problem?

A simple network with 4 hidden layers



Δb_1

$\Delta a_1 \longrightarrow \Delta z_2$

$$\Delta a_1 = \sigma'(z_1)\Delta b_1$$

$$\frac{\partial z_2}{\partial a_1} = \frac{\partial}{\partial a_1}(w_2 a_1 + b_2) = w_2$$

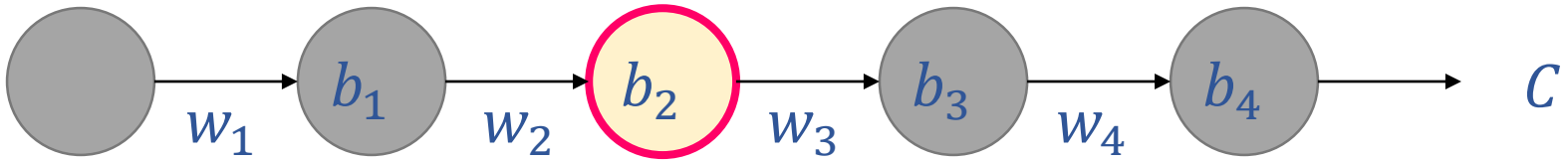
$$\frac{\Delta z_2}{\Delta a_1} \approx \frac{\partial z_2}{\partial a_1} = w_2$$

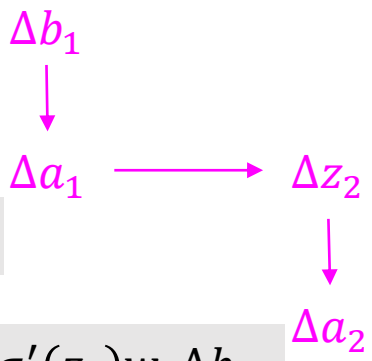
$$\Delta z_2 = \Delta a_1 w_2$$

$$\Delta z_2 = \sigma'(z_1)\Delta b_1 w_2$$

What's causing the vanishing gradient problem?

A simple network with 4 hidden layers





$$\Delta a_1 = \sigma'(z_1)\Delta b_1$$

$$\Delta z_2 = \sigma'(z_1)w_2\Delta b_1$$

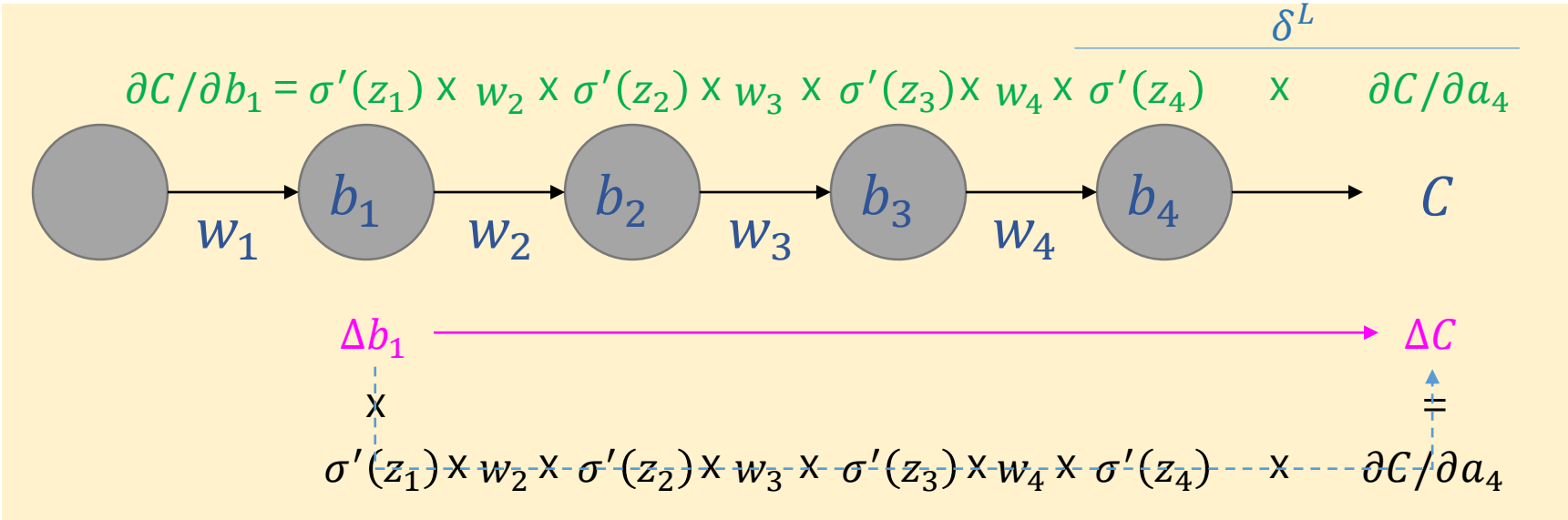
$$\frac{\partial a_2}{\partial z_2} = \frac{\partial}{\partial z_2}\sigma(z_2) = \sigma'(z_2)$$

$$\frac{\Delta a_2}{\Delta z_2} \approx \frac{\partial a_2}{\partial z_2} = \sigma'(z_2)$$

$$\Delta a_2 = \sigma'(z_2)w_2\sigma'(z_1)\Delta b_1$$

What's causing the vanishing gradient problem?

A simple network with 4 hidden layers

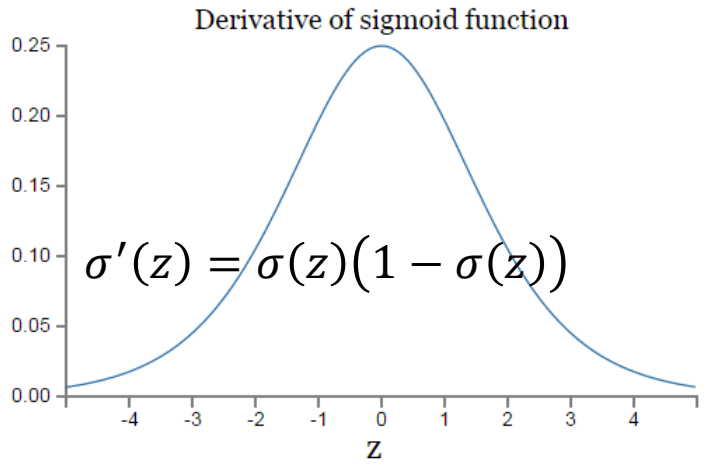


$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$

What's causing the vanishing gradient problem?

Why the vanishing gradient problem occurs?

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)w_2\sigma'(z_2)w_3\sigma'(z_3)w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}$$



Weights are generated from normalized Gaussian distribution, so weights usually satisfies |w| <1

$$|w_i\sigma'(z_j)| < 1/4$$

$$\frac{\partial C}{\partial b_3} = \sigma'(z_3)w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}$$
$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)w_2\sigma'(z_2)w_3\frac{\partial C}{\partial b_3}$$

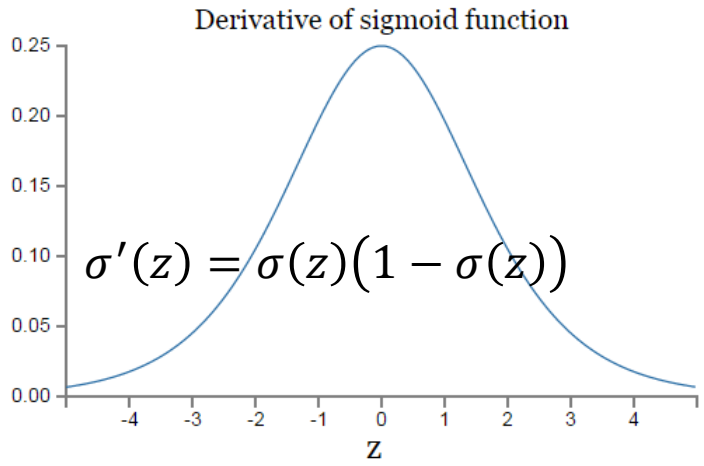
$$\left|\frac{\partial C}{\partial b_1}\right| < \frac{1}{4}\times\frac{1}{4}\times\left|\frac{\partial C}{\partial b_3}\right|$$

This is the essential origin of the vanishing gradient

What's causing the vanishing gradient problem?

The exploding gradient problem

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)w_2\sigma'(z_2)w_3\sigma'(z_3)w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}$$



Let's say |w| = 100 and the biases are chosen so that the σ′ terms are not too small (z is zero, σ′(z) = 0.25)

$$|w_i\sigma'(z_j)| \approx 25$$

$$\frac{\partial C}{\partial b_3} = \sigma'(z_3)w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}$$
$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)w_2\sigma'(z_2)w_3\frac{\partial C}{\partial b_3}$$

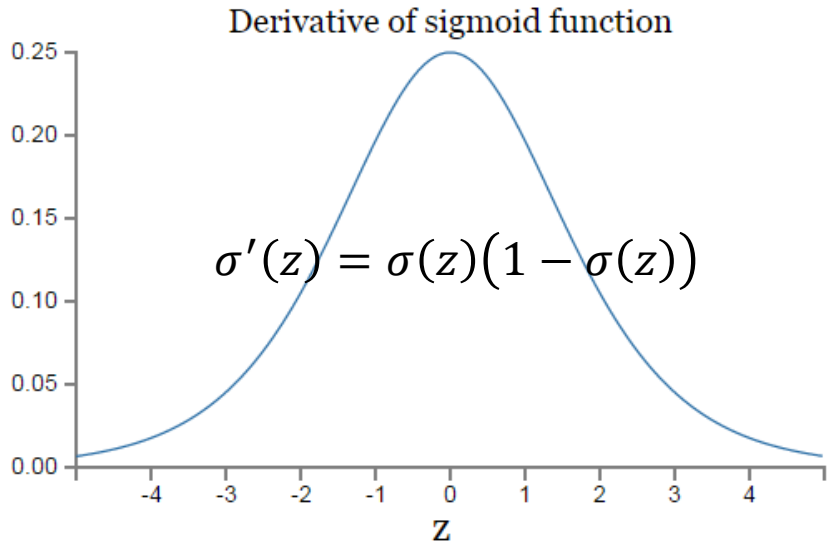
$$\left|\frac{\partial C}{\partial b_1}\right| < 25 \times 25 \times \left|\frac{\partial C}{\partial b_3}\right|$$

This is the essential origin of the exploding gradient

What's causing the vanishing gradient problem?

The prevalence of the vanishing gradient problem

$$|w\sigma'(z)| = |w\sigma'(wa+b)| \geq 1$$



Solution : very large weights (at least larger than 4)
→ it also increases 'z' that causes σ' to saturate to zero

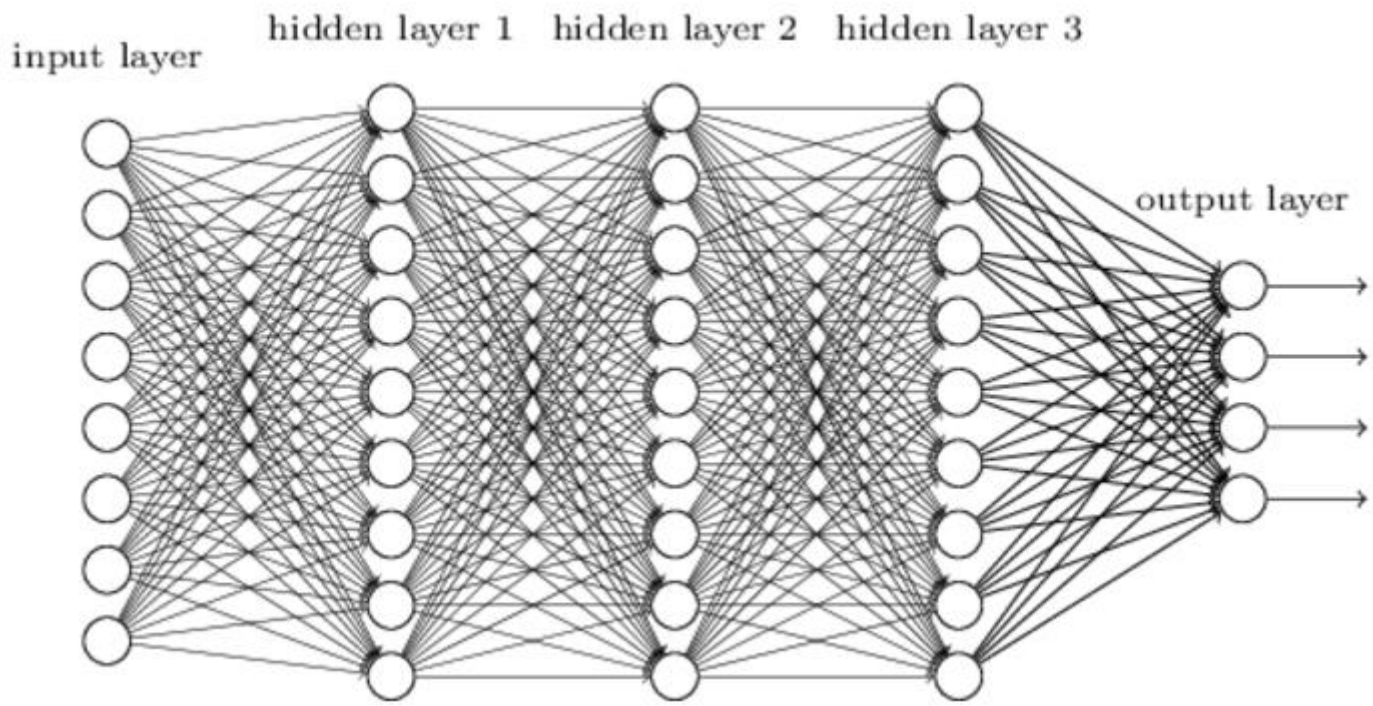
Supposing that |w|≥4, the set of 'a' satisfying that constraint can range over an interval no grater in width than

$$\frac{2}{|w|}\ln\left(\frac{|w|(1+\sqrt{1-4/|w|})}{2}-1\right)$$

The bounding the width of the range is greatest at |w|=6.9, where it takes a value = 0.45

Unstable gradients in more complex networks

Still the essential form is very similar



1. Error at the output layer

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

2. Error relationship between two adjacent layers

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

3. Gradient of C in terms of bias

$$\nabla_{b^l} C = \delta^l$$

4. Gradient of C in terms of weight

$$\nabla_{W^l} C = \delta^l (a^{l-1})^T$$

GD+BP

$$\delta^l = \Sigma'(z^l)(w^{l+1})^T \Sigma'(z^{l+1})(w^{l+2})^T \ldots \Sigma'(z^L)\nabla_a C$$

$\Sigma'(z^l)$: *a diagonal matrix whose entries are the* $\sigma'(z)$ *values for the weighted inputs to the* l*th layer*

w^l: *the weight matrices for the different layers*

Other obstacles to deep learning

Two more examples

As a first example, in 2010 Glorot and Bengio found evidence suggesting that the use of sigmoid activation functions can cause problems training deep networks. In particular, they found evidence that the use of sigmoids will cause the activations in the final hidden layer to saturate near 0 early in training, substantially slowing down learning. They suggested some alternative activation functions, which appear not to suffer as much from this saturation problem.

Understanding the difficulty of training deep feedforward neural networks, by Xavier Glorot and Yoshua Bengio (2010).

As a second example, in 2013 Sutskever, Martens, Dahl and Hinton studied the impact on deep learning of both the random weight initialization and the momentum schedule in momentum-based stochastic gradient descent. In both cases, making good choices made a substantial difference in the ability to train deep networks.

On the importance of initialization and momentum in deep learning, by Ilya Sutskever, James Martens, George Dahl and Geoffrey Hinton (2013).