1DT109 - Accelerating systems with FPGAs
Unstable gradients

Riccardo De Masellis

Uppsala University

December 6, 2021

Recap

Universality theorem says that one hidden layer is enough to approximate any continuous function. However:

In many tasks, e.g., visual pattern recognition deep networks are a better choice.

Sometimes having multiple layers reduces the number of total neurons[1].

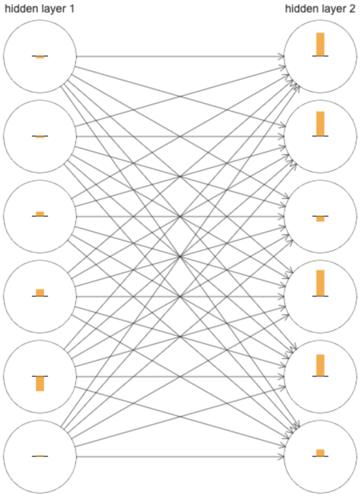
[1]Something similar happens with circuits depth as well.

How do we train deep networks?

Problem

Layers learn at different speeds. In particular, early layers barely change their weights at all (but it could happen the other way around as well).

The vanishing gradient problem



Network [748, 30, 30, 10], showing only the top six neurons of the two hidden layers.

Bars represent $\frac{\partial C}{\partial b}$ for each neuron, on the MNIST digits, at the beginning of the training.

Coincidence? Don't think so...

- $\delta_j^\ell = \frac{\partial C}{\partial b_j^\ell}$, "gradient"[2] of j-th neuron in ℓ-th layer;
- δ^ℓ vector of all "gradients" of the ℓ-th layer;
- with $\left\|\delta^\ell\right\| = \sqrt[2]{\delta_1^\ell + \ldots + \delta_n^\ell}$, with n number of neuron in layer ℓ, we denote the "speed" of learning of layer ℓ.

[2] not really, only for the bias

Coincidence? Don't think so...

- $\delta^\ell_j = \frac{\partial C}{\partial b^\ell_j}$, "gradient"[2] of j-th neuron in ℓ-th layer;
- δ^ℓ vector of all "gradients" of the ℓ-th layer;
- with $\left\|\delta^\ell\right\| = \sqrt[2]{\delta^\ell_1 + \ldots + \delta^\ell_n}$, with n number of neuron in layer ℓ, we denote the "speed" of learning of layer ℓ.

On the example above we get:

- $\left\|\delta^1\right\| = 0.07$
- $\left\|\delta^2\right\| = 0.31$

[2] not really, only for the bias

Coincidence? Don't think so...

- $\delta_j^\ell = \frac{\partial C}{\partial b_j^\ell}$, "gradient"[2] of j-th neuron in ℓ-th layer;
- δ^ℓ vector of all "gradients" of the ℓ-th layer;
- with $\left\|\delta^\ell\right\| = \sqrt[2]{\delta_1^\ell + \ldots + \delta_n^\ell}$, with n number of neuron in layer ℓ, we denote the "speed" of learning of layer ℓ.

On the example above we get:

- $\left\|\delta^1\right\| = 0.07$
- $\left\|\delta^2\right\| = 0.31$

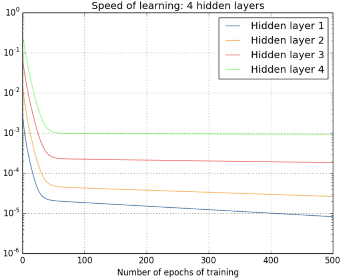
With a $[784, 30, 30, 30, 10]$ we get:

- $\left\|\delta^1\right\| = 0.012$
- $\left\|\delta^2\right\| = 0.060$
- $\left\|\delta^3\right\| = 0.283$

[2] not really, only for the bias

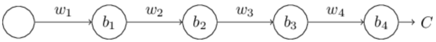
Does training change things?

Batch gradient descent, 1000 images, 500 epochs.



- The speed of learning start slower in earlier layers;
- it keeps being slower during the training (100 ratio).

Let's analyse a simple network...

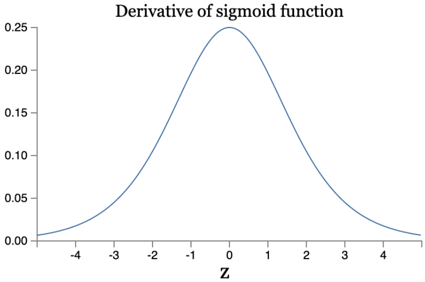


Let's analyse $\frac{\partial C}{\partial b_1}$:

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

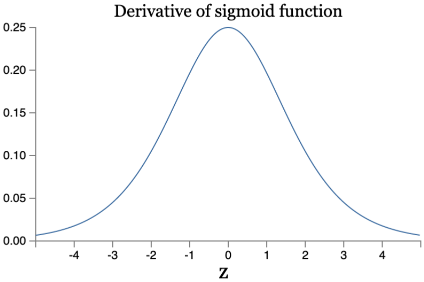
- $a_j = \sigma(z_j)$
- $z_j = w_j a_{j-1} + b_j$

...carefully!



- Maximum is at $\sigma'(0) = \frac{1}{4}$;
- $|w_j| < 1$ because they are initialised according to a Gaussian with mean 0 and standard deviation 1.

...carefully!



- Maximum is at $\sigma'(0) = \frac{1}{4}$;
- $|w_j| < 1$ because they are initialised according to a Gaussian with mean 0 and standard deviation 1.

$$|w_j \sigma'(z_j)| < \frac{1}{4}$$

Vanishing gradient explained!

The more such terms are multiplied, the smallest the product becomes.

The exploding gradient

If, during the learning, it happens that terms $|w_j \sigma'(z_j)|$ are (much) larger than 1, then multiplied together we get an exploding gradient.

However, this does not happen often in practice.

How to cope with unstable gradient

Again, it's kind of heuristic:

- For image recognition, using convolutional neural network helps;
- using rectified linear activation functions *usually* speeds up the training;
- initialise the weights to neurons as Gaussian random variables with mean 0 and standard deviation $\frac{1}{\sqrt{n_{in}}}$ where n_{in} is the number of inputs of the neuron.