Published in Towards Data Science

Chi-Feng Wang Follow

Jan 8, 2019 · 3 min read · Listen

The Vanishing Gradient Problem

The Problem, Its Causes, Its Significance, and Its Solutions

Title Image // Source

The problem:

Why:

Certain activation functions, like the sigmoid function, squishes a large input space into a small input space between 0 and 1. Therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small.

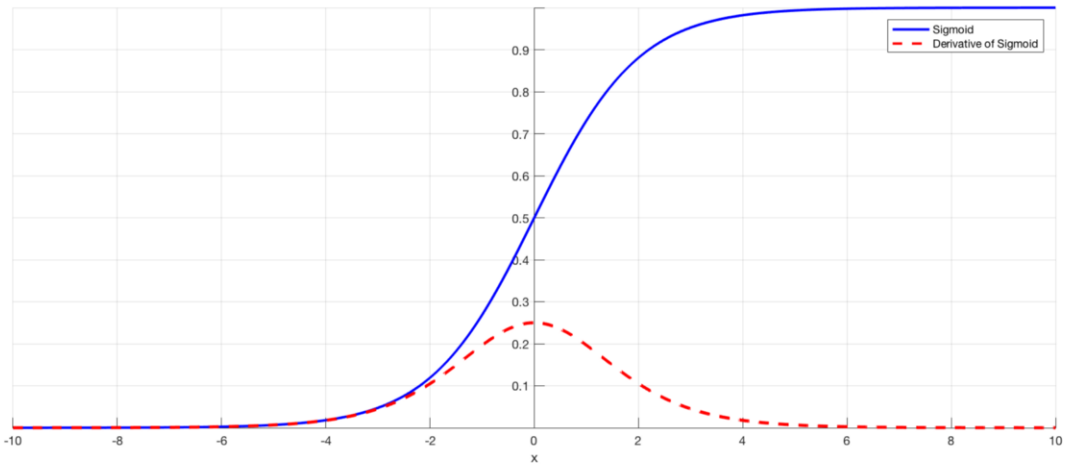

Image 1: The sigmoid function and its derivative // Source

As an example, Image 1 is the sigmoid function and its derivative. Note how when the inputs of the sigmoid function becomes larger or smaller (when |x| becomes bigger), the derivative becomes close to zero.

Why it's significant:

For shallow network with only a few layers that use these activations, this isn't a big problem. However, when more layers are used, it can cause the gradient to be too small for training to work effectively.

Gradients of neural networks are found using backpropagation. Simply put, backpropagation finds the derivatives of the network by moving layer by layer from the final layer to the initial one. By the chain rule, the derivatives of each layer are multiplied down the network (from the final layer to the initial) to compute the derivatives of the initial layers.

propagate down to the initial layers.

A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training session. Since these initial layers are often crucial to recognizing the core elements of the input data, it can lead to overall inaccuracy of the whole network.

Solutions:

The simplest solution is to use other activation functions, such as ReLU, which doesn't cause a small derivative.

Residual networks are another solution, as they provide residual connections straight to earlier layers. As seen in Image 2, the residual connection directly adds the value at the beginning of the block, **x**, to the end of the block ($F(x)+x$). This residual connection doesn't go through activation functions that "squashes" the derivatives, resulting in a higher overall derivative of the block.

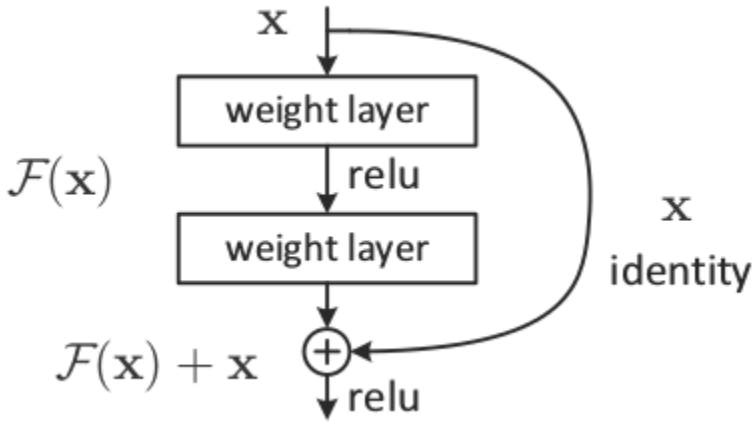

Image 2: A residual block

Finally, batch normalization layers can also resolve the issue. As stated before, the problem arises when a large input space is mapped to a small one, causing the derivatives to disappear. In Image 1, this is most clearly seen at when $|x|$ is big. Batch normalization reduces this problem by simply normalizing the input so $|x|$ doesn't

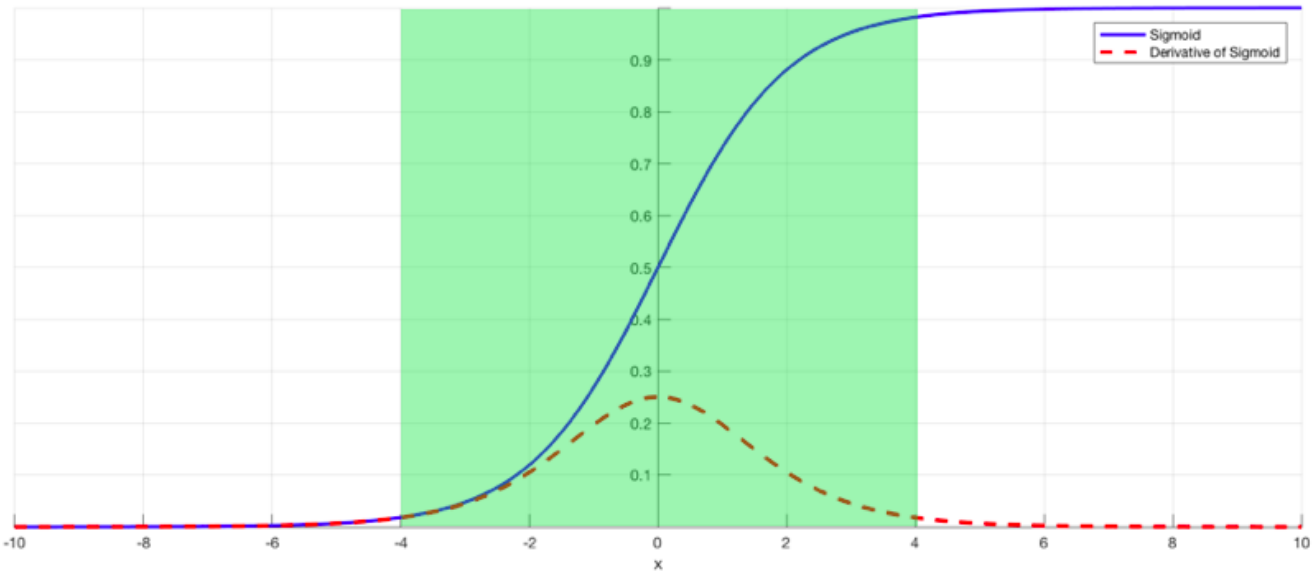

Image 3: Sigmoid function with restricted inputs

Do leave a comment below if you have any questions or suggestions :)

Read these articles for more information:

- https://www.quora.com/What-is-the-vanishing-gradient-problem
- https://en.wikipedia.org/wiki/Vanishing_gradient_problem
- https://towardsdatascience.com/intuit-and-implement-batch-normalization-c05480333c5b