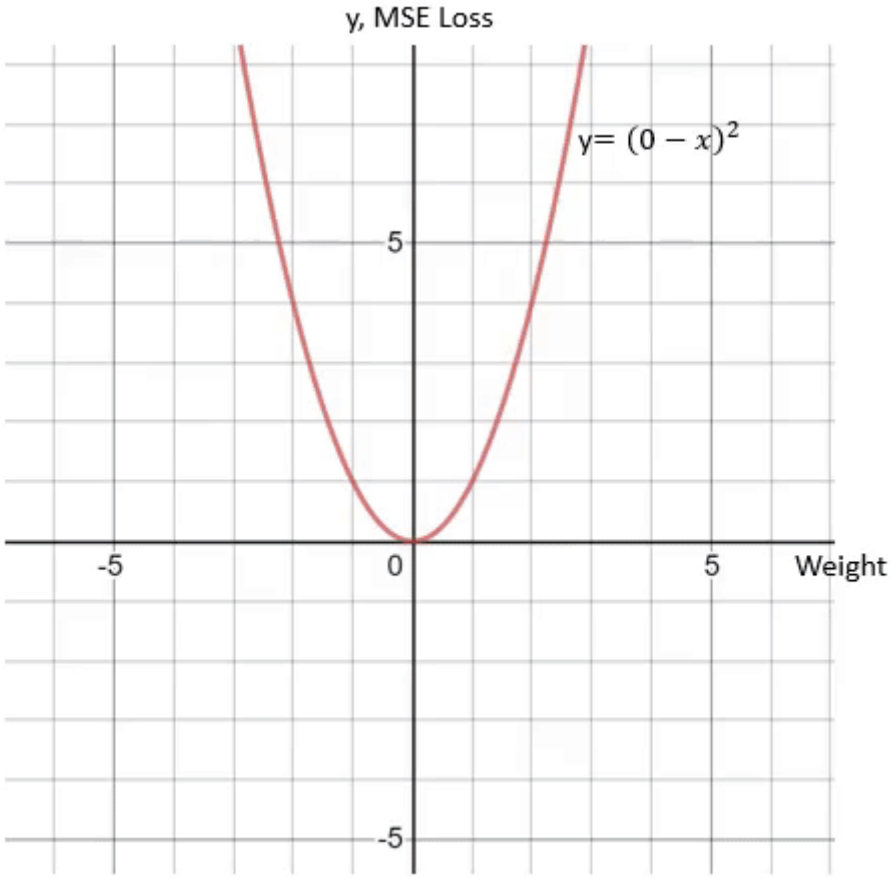
Visualizing Relationships between Loss Functions and Gradient Descent

Gradient Descent working on Loss/Activation functions

Hugegene

Apr 26 · 4 min read ★

This article assumes you have prior knowledge on training Neural Net and seeks to demystify the relationships between Loss function, Gradient Descent and Backpropagation through visualisation.



The figure on the left shows the relationship between a loss function and gradient descent.

To visualise gradient descent, imagine an example that is over-simplified to a neural net's last node outputting a weight number, w, and the target is 0. The Loss function, in this case, is Mean Square Error (MSE).

The derivative of MSE, dy/dw, is positive when w is bigger than 0. A positive dy/dw can be interpreted as a positive step in w will result in a positive change in y. To lead to a decrease in loss, a negative step in the w direction is needed:

$$w' = w - \left(+\frac{dy}{dw}\right)$$

When w is smaller than 0, dy/dw of MSE is negative and it implies that a positive step in w will result in a negative change in y. To lead to a decrease in loss, a positive step in the w direction is needed:

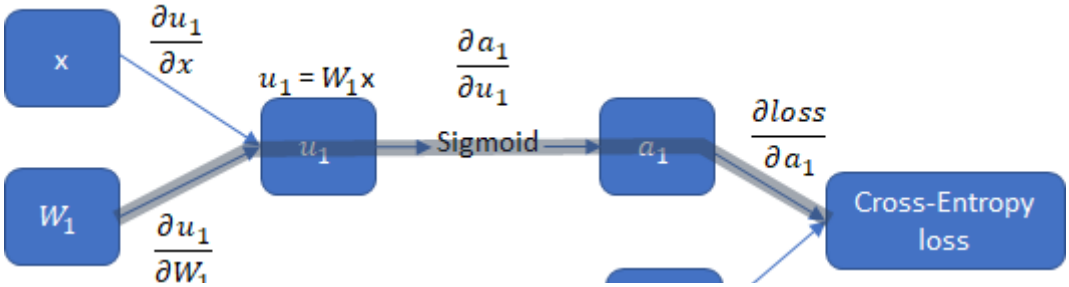
$$w' = w - \left(-\frac{dy}{dw}\right)$$

Hence the formula that summarized the update of weight is as follows:

$$w' = w - learning\_rate * \frac{\partial Loss}{\partial w}$$

where learning_rate is a constant that adjusts how much percentage of the derivative should be stepped. Learning_rate should be adjusted to prevent w from taking too small a step or too big a step. Learning_rate should also be adjusted to prevent gradient explosion (too big a gradient) or vanishing gradient problem (too small a gradient).

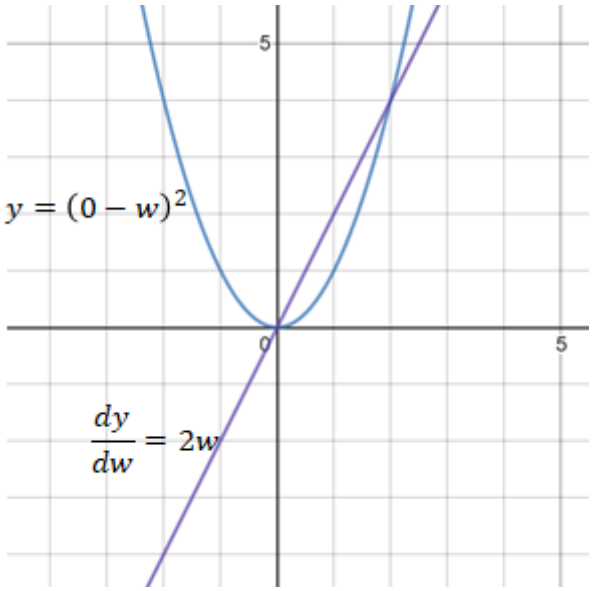
For a longer and more realistic computation graph where the weight is behind a sigmoid activation, to update the weight w1, the derivative of loss with respect to w1 can be found as follows:



$$\frac{\partial loss}{\partial W_1} = \frac{\partial loss}{\partial a_1} \times \frac{\partial a_1}{\partial u_1} \times \frac{\partial u_1}{\partial W_1}$$

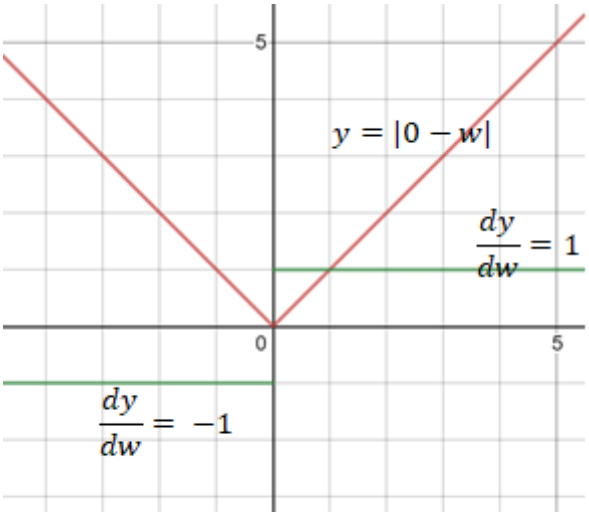
the derivative of loss with respect to weight, w1

As seen from the illustrated steps above, the weight in the neural net is revised or backpropagated by the derivative of the Loss function and not by the loss function. The loss function does not contribute to the backpropagation.



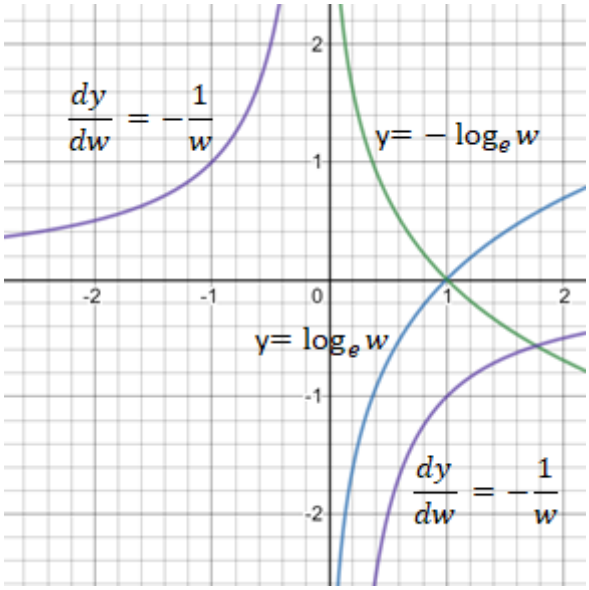
L2 Loss

The derivative of MSE (L2 loss) has a magnitude of 2w. The changes in the magnitude of the derivative of MSE help to backpropagate a bigger step to w when w is far away from target 0 and smaller step to w when w is closer to target 0



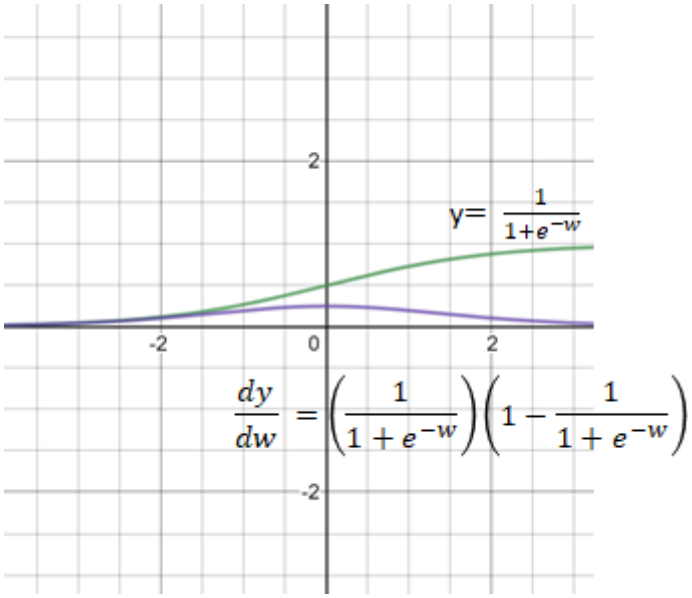
L1 Loss

Mean Absolute Error (MAE) (L1 loss) has a constant derivative of 1 or negative 1, which may not be ideal in differentiating how far is w from the target. Backpropagation only uses the derivative of the loss function and does not use the loss function.



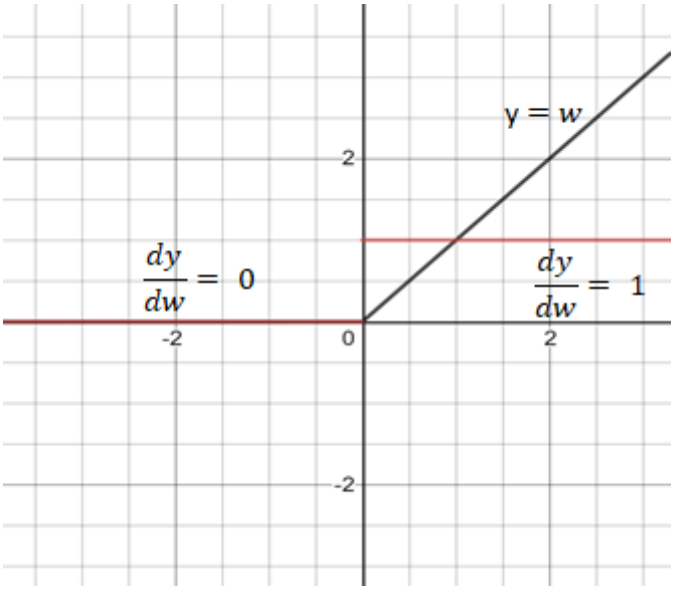
Cross-Entropy Loss

Cross-Entropy loss only concerns the domain of w in between 0 and 1. As w approaches 1, Cross-Entropy decrease to 0. The derivative of Cross-Entropy is -1/w.



Sigmoid activation

Sigmoid function has a derivative of range in between 0 and 0.25. Multiple products of the derivatives of sigmoid functions may result in a very small number close to 0 and this makes backpropagation useless. This is known as the vanishing gradient problem.



Relu activation

Relu makes a good activation function as the derivative is 1 or 0, allowing a constant update to the weights or 0 updates to the weights in backpropagation.

Gradient Descent Backpropagation Loss Activation Functions Deep Learning

About Help Legal