Problem Set 3 – Loss Functions and Fitting Models

DS542 – DL4DS

Spring, 2025

Note: Refer to the equations in the *Understanding Deep Learning* textbook to solve the following problems.

Problem 5.9

Consider a multivariate regression problem in which we predict the height of an individual in meters and their weight in kilos from some data x. Here, the units take quite different values. What problems do you see this causing? Propose two solutions to these problems.

0.1 Solution

The problem is that both of the variables we want to predict follow different distributions and have different range of values. In multivariate prediction we assume that the variables are independent, but in the case of weight and height, they are not. For example, if we have a very tall person, we can expect that their weight will be higher than the average. Therefore their error is also not independent because they are correlated. This multicollinearity affects the performance of the model and reduces the precision. One solution could be to normalize both variables to be within a range of 0 and 1. Other could be to use a different model that can handle multicollinearity, like a neural network. Because neural networks can learn the correlation between the variables and adjust the weights accordingly.

Problem 6.6

Which of the functions in Figure 6.11 from the book is convex? Justify your answer. Characterize each of the points 1–7 as (i) a local minimum, (ii) the global minimum, or (iii) neither.

0.2 Solution

A function is convex if its second derivative is positive. Furthermore, if the function is convex, it should look like a bowl; if we draw a line between two points of the function, the function should lie below the line. Therefore, there is only one convex function, and that is **B**. The points are characterized as follows:

1. Neither
2. Global minimum

Problem 6.10

Show that the momentum term m_t (equation (6.11)) is an infinite weighted sum of the gradients at the previous iterations and derive an expression for the coefficients (weights) of that sum.

0.3 Solution

We start with the momentum update formula:

$$m_{t+1} = \beta m_t + (1 - \beta) g_t \tag{1}$$

where:

- m_t is the momentum term at iteration t
- β is the momentum coefficient
- g_t is the gradient at iteration t

Expanding m_t recursively:

$$m_t = \beta m_{t-1} + (1 - \beta) g_{t-1} \tag{2}$$

Substituting m_{t-1}:

$$m_{t-1} = \beta m_{t-2} + (1 - \beta) g_{t-2} \tag{3}$$

Continuing this expansion back to $t = 0$, assuming $m_0 = 0$:

$$m_{t+1} = (1 - \beta) g_t + \beta(1 - \beta) g_{t-1} + \beta^2 (1 - \beta) g_{t-2} + \cdots + \beta^t (1 - \beta) g_0 \tag{4}$$

This can be rewritten:

$$m_{t+1} = \sum_{k=0}^{t} \beta^k (1 - \beta) g_{t-k} \tag{5}$$

This equation shows that the momentum term is an exponentially weighted moving average of past gradients, where recent gradients have higher weights due to the β^k term decreasing as k increases.

The weight for each past gradient g_{t-k} is given by:

$$w_k = \beta^k(1-\beta) \tag{6}$$

The sum of these weights is:

$$\sum_{k=0}^{\infty} w_k = (1-\beta)\sum_{k=0}^{\infty}\beta^k = (1-\beta)\frac{1}{1-\beta} = 1 \tag{7}$$

This confirms that the gradient contributions form a proper weighted sum. The momentum term m_t is an infinite weighted sum of past gradients, with exponentially decreasing weights given by $\beta^k(1-\beta)$. The larger β is, the more influence past gradients have, smoothing out the updates and accelerating convergence in deep learning optimizers like SGD with momentum.