Deep Learning

Synthesis Questions

Introduction to Neural Network

1. Define the following words:

- Neuron
- Layer
- Hidden Layer
- Weight
- Bias

2. Calculate the dimension of...

You have a neural network with an input dimension of 3, a hidden dimension of 4, and an output dimension of 1. Calculate the...

- Dimension of W between the input and hidden layers: $W \in \mathbb{R}^{4\times3}$
- Dimension of W between the hidden and output layers: $W \in \mathbb{R}^{4\times1}$
- Dimension of b for the hidden layer: $b \in \mathbb{R}^{4\times1}$

3. Why can't we use a linear classifier to solve the XOR problem?

Because the a linear classifier draws a boundary, while the XOR classification problem is not linearly separable.

Non-Linearity and Activation Functions

At first, each node has a raw output value before the activation function is applied. **Activation functions** are applied to each node, giving non-linearity to the models to solve complex problems. They play a crucial role in building a desired model; they affect range of outputs, deciding if the model will solve a regression or classification problem. According to the *Universal Approximation Theorem*, activation functions also allow a neural network to approximate any continuous function at a desired level of accuracy.

Backpropagation

1. What is the purpose of a loss function?

Loss function compares the expected value to the predicted value to find out how big the difference is. After getting the value from the loss function, we start backpropagation, where we use optimizers to adjusts the weights accordingly to minimize the value of this loss function.

2. What are the different use cases for Mean Squared Error vs. Cross Entropy Loss?

Mean Squared Error function is mostly used for numerical regression tasks, like predicting a house price. On the other hand, Cross Entropy Loss is used for classification task, like figuring out how likely would this picture be showing a cat instead of a dog.

3. What is the difference between a derivative and a gradient?

While derivative is for **single-variable** functions, gradient is for **multi-variable** functions like the neural network. It is a vector containing values of all partial derivatives with respect to **all** inputs

4. Find the gradient for this function at (3, 2, 1):

$$f(x,y,z) = \frac{2}{3}x^2 + y^2 - 2y + z^4 - \frac{4}{3}z^3$$

$$\frac{\partial f}{\partial x} = \frac{4}{3}x$$

$$\frac{\partial f}{\partial y} = 2y - 2$$

$$\frac{\partial f}{\partial z} = 4z^3 - 4z^2$$

$$\nabla f(x,y,z) = \begin{bmatrix} 4 \\ 2 \\ 0 \end{bmatrix}$$

5. Find the derivative of the sigmoid (σ) function and plot it.

Why might this activation function prevent gradients from flowing back through the network during the backward pass?

$$\sigma^{'}(x) = \frac{1}{1+e^{-x}}$$

$$\sigma^{'}(x) = \sigma(x)(1-\sigma(x))$$

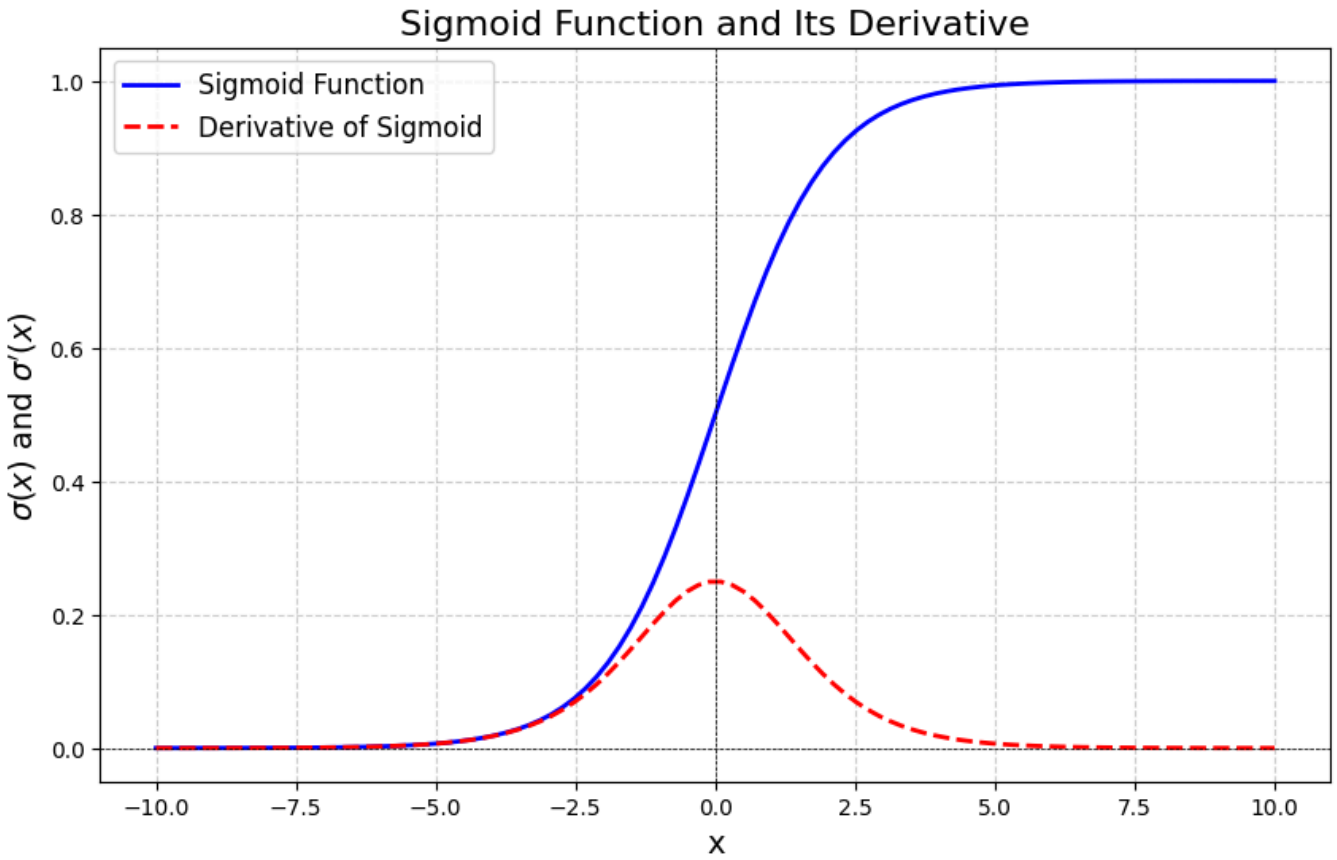

image-130.png

Sigmoid squashes values too strongly near 0 or 1. In those saturated regions, where the x is very large positive or negative value, its derivative is basically ~0. During backpropagation, gradients get multiplied by this small derivative repeatedly, so they shrink dramatically as they move backward through layers. As a result, earlier layers barely get gradients and can't learn properly.

6. Why might the "steps" taken by SGD not be as directly in the most optimal direction compared to GD?

SGD is making decisions based on tiny bits of data at a time called the mini-batch, so each step may be noisy, like a little off or rotated. That noise means the step direction might not line up perfectly with the optimal downhill direction like GD would. It's faster and cheaper to compute, but the direction to minimize the loss is less accurate.

Regularization

1. In your own words, why does Dropout work?

Dropout works because it stops the model from relying too much on specific neurons. By randomly shutting off neurons during training, the network is forced to spread learning across many pathways. This makes the model more flexible and generalizable, reducing overfitting. It's kind of like training the "weaker muscles" on purpose so every part of the network becomes useful instead of just a few strong ones.

2. Why do you think a moving average of μ and σ are useful for a network with batch normalization layers if there are many batches to process?

It keeps everything centered and stable over time. Instead of letting every random mini-batch shift things around, a moving average makes a long-term memory of the mean and variance. So at the end, the network uses those stable values instead of noisy batch-by-batch statistics, which makes the model way more reliable.

3. Think of, or find online, a method of regularization within neural networks not discussed here. Write two to three sentences on how it works

L1 regularization adds a penalty to the loss based on the absolute value of each weight, which encourages many weights to go exactly to zero. This forces the network to become sparse, reducing overfitting and helping the model focus on only the most meaningful connections.