David Pranata Timothy Nangoi

300604132

Nangoidavi

Exercise 1: Single-Layer Network and Backpropagation

In this exercise, we worked with a single-layer neural network and implemented backpropagation to optimize its weights and biases. The goal was to observe how the network's convergence behavior changes when using different target values and understand the limitations of a single-layer architecture in handling non-linearly separable data.

Here's a step-by-step breakdown of the exercise:

1. **Neural Network Setup**: we began with a simple single-layer neural network. The network had two input neurons (representing features), one output neuron (for prediction), and a sigmoid activation function. The network's output was calculated using the formula:
   makefile

```
y = σ(bias + w0*x0 + w1*x1)
```

where σ(z) is the sigmoid activation function and bias, w0, and w1 are the bias and weights of the neuron, respectively.
Backpropagation Formulas: we were provided with formulas to calculate the gradients (slopes) of the loss function with respect to the bias and weights. These gradients were used to update the bias and weights during the training process. The key formulas were:

- Calculate δ (delta):
  scss

```
δ = σ(z) * (1 - σ(z)) * (y - t)
```

where t is the target value.
Calculate gradients:
bash

```
dE/dw0 = δ * x0 * 2
dE/dw1 = δ * x1 * 2
dE/dbias = δ * 2
```

   ○

2. **Training Data Modification**: The training data set was modified to include non-linearly separable data points. This was done by changing the target values to create a more complex pattern.
3. **Training and Convergence**: Using the modified data set, we trained the single-layer neural network with the backpropagation algorithm. The goal was to observe the

convergence behavior of the network and see if it could accurately fit the new data distribution.

4. **Observations**: Through this exercise, we learned that a single-layer neural network struggles to handle complex patterns that are not linearly separable. The network might converge to relatively high error values or fail to converge altogether. This demonstrates the limitations of a shallow architecture in capturing intricate relationships in the data.

In summary, Exercise 1 highlighted the challenges faced by single-layer neural networks when dealing with non-linearly separable data. It showcased the importance of deeper architectures for capturing more complex patterns and paved the way for further exploration of multi-layer networks in subsequent exercises.

Exercise 2: Effect of Changing Target Values

In this exercise, we investigated how changing the target values of specific data points in the training set impacts the training process and convergence behavior of a neural network. The goal was to observe and analyze the differences in training dynamics and convergence graphs when the target values are modified.

Of course this exercise will result in nothing as it has no way of drawing a line between two intersecting vectors without them colliding or standing on top of another.

Exercise 3: Two-Layer Network with Same Weights

Loss of Representation Power: The strength of neural networks lies in their ability to capture and represent diverse features and patterns in the data. When hidden layer neurons share the same weights, they essentially duplicate each other's functionality. This loss of diversity and representation power severely limits the network's capacity to learn and represent complex relationships in the data.

Exercise 4: Tuning Initial Values Deviation

Changing Deviation: Instead of initializing all hidden layer neuron weights to be the same, we varied the deviation of the initial weight values. The deviation controlled the spread or diversity of the initial weights. Lower deviations produced more similar initial weights, while higher deviations introduced more diversity.