Assignment 1: Logistic Regression with Varying Batch Sizes
Deep learning - EE569

Talal Malek Badi - 2200208609
Supervisor Dr. Nuri Benbarka

November 29, 2024

Contents

List of Figures

1 Introduction

This Assignment includes implementing logistic regression using a custom computational graph framework, modifying the model to include batching, and investigating the effect of batch size on training and decision boundaries. Each task's corresponding results are documented below and the python implementation is in git-hub repository, Each task has it's own commit in the repository for clean version control.

—

2 Task 1: Implementing a Linear Computation Node

2.1 Task Description

The task required creating a custom computation node, `Linear`, that performs the following operation during the forward pass:

$$\mathbf{y} = \mathbf{A} \cdot \mathbf{x} + \mathbf{b},$$

where:

- $\mathbf{A}$: Weight matrix,
- $\mathbf{x}$: Input vector,
- $\mathbf{b}$: Bias vector.

The backward method calculates the gradients with respect to all inputs.

2.2 Results

The `Linear` node was successfully implemented in EDF. Gradients were verified using backpropagation,

2.3 implementation

GitHub commit for task 1

—

3 Task 2: Integrating the Linear Node into Logistic Regression

3.1 Task Description

The `Linear` node replaced the `Multiply` and `Addition` nodes in the logistic regression code. The forward pass calculates the linear node, and the sigmoid activation calculate the probabilities.

3.2 Results

The updated code successfully computed the linear node and probabilities using the `Linear` node. The loss function and gradient computation still works as expected, and the framework demonstrated accurate forward and backward propagation.

3.3 implementation

GitHub commit for this task 2

4 Task 3: Introducing Batching

4.1 Task Description

Batching was implementd to process multiple data points. Instead of a single input vector, a batch of input vectors was passed, and the bias vector was adjusted accordingly. Batch sizes tested include $B \in \{1, 2, 4, 8, 16, 32, 64, 128\}$.

4.2 Results

Batches improved computational efficiency, as multiple samples were processed in parallel

4.3 implementation

GitHub commit for this task 3&4

5 Task 4: Investigating the Effect of Batch Size

5.1 Task Description

The effect of varying batch sizes on the training process was investigated. Training loss and decision boundaries were analyzed for each batch size.

5.2 Results

The decision boundaries for each batch size are shown below:

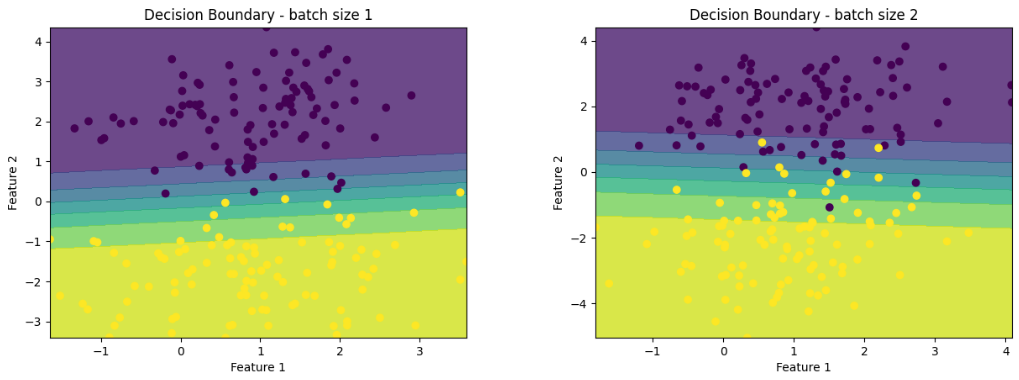

Figure 1: Decision Boundary for Batch Size = 1 & 2

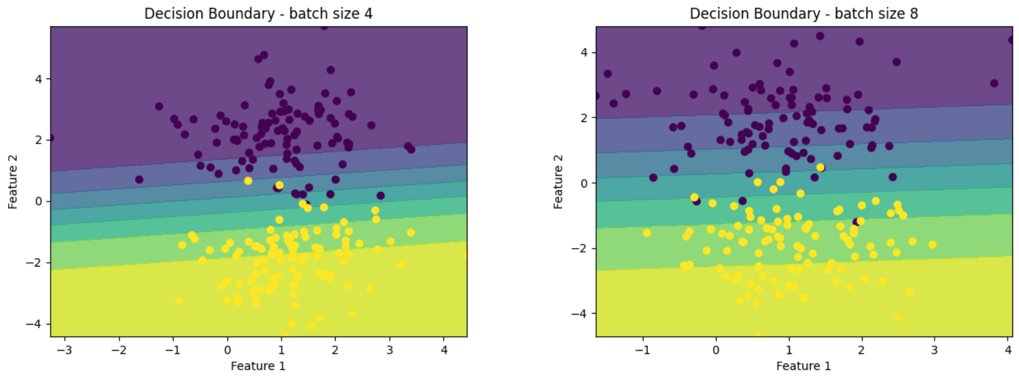

Figure 2: Decision Boundary for Batch Size = 4 & 8

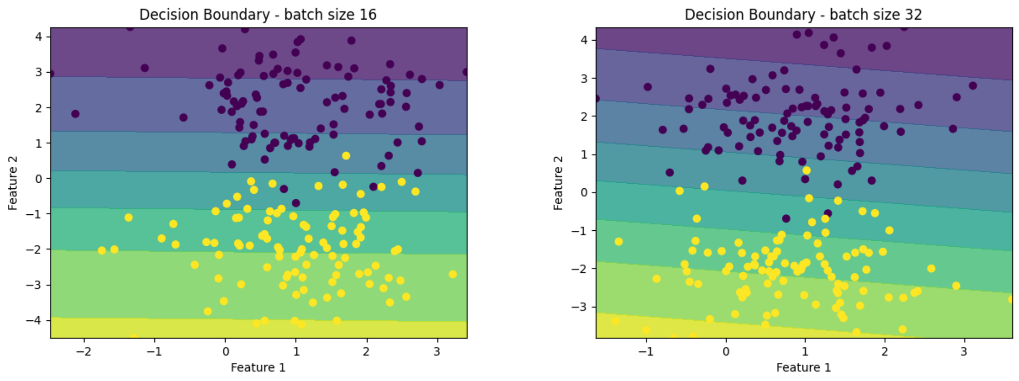

Figure 3: Decision Boundary for Batch Size = 32 & 16

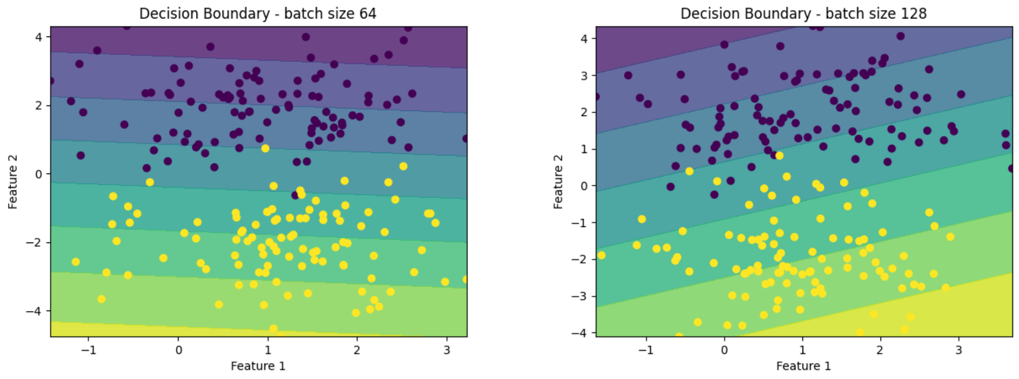

Figure 4: Decision Boundary for Batch Size = 64 & 128

Observations:

- **Small Batch Sizes (1, 2):** High convergence in decision boundaries due to the effect of each point in small batch (which makes it more sensitive to noise).
- **Moderate Batch Sizes (4, 8):** results in smoother decision boundaries.
- **Large Batch Sizes (16, 32, 64):** even smoother
- **Batch Size = 128:** showing minimal noise but slower convergence.

below figure show high batch (128) which has slower learning but faster computation , using 10000 ephocs, we found great and fast output.

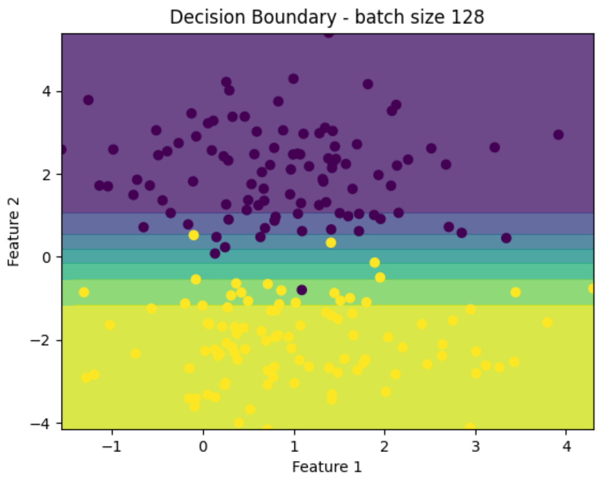

Figure 5: Decision Boundary for Batch Size = 128 and ephoc 10000

we can say that we can bench mark between ephoc and batch size, for large batch size it is better to use higher ephoc because of less convergence in larger batch sizes.

6 Conclusion

In this assignment, we added a special `Linear` node to a logistic regression model and tested different batch sizes to see how they affect the model's learning. We found that smaller batches learn quickly but are very sensitive to changes in data. On the other hand, larger batches are more stable but learn slower and need more training rounds (epochs) to perform well. This work shows that choosing the right batch size depends on what you need from the model and the resources you have. It's a balance between how fast you want the model to learn and how accurately you want it to perform