Banana Dataset

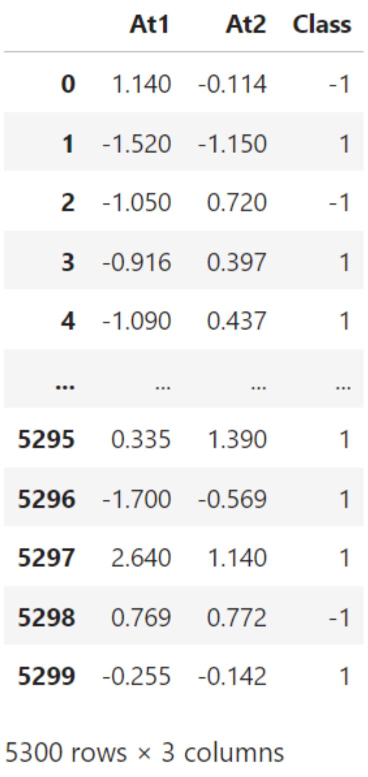
K S Ananth
Task 2

1 Overview:

Banana Dataset is a classic binary classification problem. It consists of two features and two classes that form banana-shaped clusters in a two-dimensional space. The Banana Dataset consists of 5300 input values. Each input values contains two features values which marks the x and y axis point in the plot. The output class is one of 1 or -1. We will use back-propagation algorithm to train.

2 Implementation Before Training:

2.1 Dataset:



	At1	At2	Class
0	1.140	-0.114	-1
1	-1.520	-1.150	1
2	-1.050	0.720	-1
3	-0.916	0.397	1
4	-1.090	0.437	1
...	...	...	...
5295	0.335	1.390	1
5296	-1.700	-0.569	1
5297	2.640	1.140	1
5298	0.769	0.772	-1
5299	-0.255	-0.142	1

5300 rows × 3 columns

Figure 1: Dataset

2.2 Dataset Plotting:

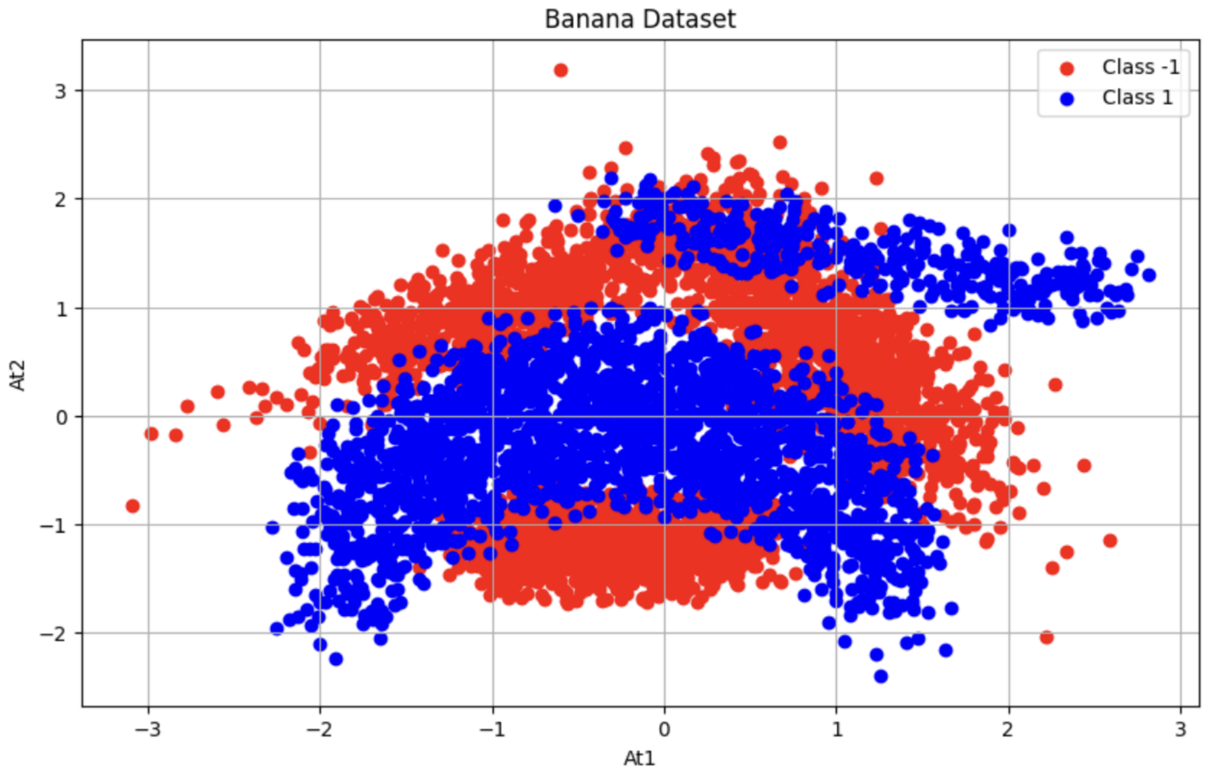


Figure 2: Dataset Plotting

2.3 Dataset Division:

Now, let us divide the dataset into 2 parts one for training the neural network and another to test the trained neural network. Let us divide the total of 5300 inputs into training set with roughly 400 inputs and the testing set having 4900 inputs. We can do this by doing this: In this training process we have considered 402 training inputs and the rest is further divided to validation data and test data. Wherein validation data has about 1028 inputs and the rest is for testing. The use of validation data will be seen later in this report.

3 Training:

3.1 Train:

Let us use the same back propagation algorithm given below with MSE loss function and use tanh as the activation function. Let the learning rate be 0.1 as learning rates 0.001 and 0.01 are too small for this dataset. Let us run the algorithm for 200 iterations/epochs. Also, print the loss values at each iteration which can help us further analyse. Refer algorithm 1 to see the back propagation used.

Algorithm 1 Back Propagation Algorithm

```
    Input: Training data D = $\{(x_i, y_i)\}_{i=1}^{n}$
 1: Initialize network using Gaussian distribution with mean 0 and variance 1 to all weights and biases
 2:     $w_{ij} \leftarrow$ a small number,     $b_j \leftarrow$ small number
 3: for each Epoch: do
 4:   for each training example (x, y) ∈ D do
 5:     /* Propagate inputs forward to find outputs */
 6:     for each neuron i in input layer: do
 7:       $a_i^0 \leftarrow x_i$
 8:     end for
 9:     for layer l=1 to L: do
10:       for each neuron i in layer l: do
11:         $z_i^l \leftarrow \sum_i w_{ij}^l a_j^{l-1} + b_j^l$          {j ∈ {1, 2, ..., (neurons_layer(l-1))}}
12:         $a_i^l \leftarrow f_l(z_i^l)$
13:       end for
14:     end for
15:     /* Propagate deltas backwards from output layer to input layer */
16:     for each neuron i in output layer: do
17:       $\delta_i^L \leftarrow \frac{(\hat{y}_i - a_i^L)}{n} f_L'(z_i^L)$
18:       $\frac{\partial C_x}{\partial w_{ik}^L} \leftarrow \delta_i^L a_k^{L-1}, \quad \frac{\partial C_x}{\partial b_i^L} \leftarrow \delta_i^L$
19:       $\frac{\partial C}{\partial w_{ik}^L} \leftarrow \frac{(n\frac{\partial C}{\partial w_{ik}^L} + (\hat{y}_x - y_x)\frac{\partial C_x}{\partial w_{ik}^L})}{n}, \quad \frac{\partial C}{\partial b_i^L} \leftarrow \frac{(n\frac{\partial C}{\partial b_i^L} + (\hat{y}_x - y_x)\frac{\partial C_x}{\partial b_i^L})}{n}$
20:     end for
21:     for layer l=L-1 to 1: do
22:       for each neuron i in layer l: do
23:         $\delta_i^l \leftarrow f_l'(z_i^l) \sum_j \delta_j^{l+1} w_{ji}^{l+1}$          {j ∈ {1, 2, ..., (neurons_layer(l+1))}}
24:         $\frac{\partial C_x}{\partial w_{ik}^l} \leftarrow \delta_i^l a_k^{l-1}, \quad \frac{\partial C_x}{\partial b_i^l} \leftarrow \delta_i^l$
25:         $\frac{\partial C}{\partial w_{ik}^l} \leftarrow \frac{(n\frac{\partial C}{\partial w_{ik}^l} + (\hat{y}_x - y_x)\frac{\partial C_x}{\partial w_{ik}^l})}{n}, \quad \frac{\partial C}{\partial b_i^l} \leftarrow \frac{(n\frac{\partial C}{\partial b_i^l} + (\hat{y}_x - y_x)\frac{\partial C_x}{\partial b_i^l})}{n}$
26:       end for
27:     end for
28:   end for
29:   for all l,j,k: do
30:     /* Update values of weights and biases using gradient descent */
31:     $w_{jk}^l \leftarrow w_{jk}^l - \eta\frac{\partial C}{\partial w_{jk}^l}, \quad b_j^l \leftarrow b_j^l - \eta\frac{\partial C}{\partial b_j^l}$
32:   end for
33: end for
```

4 Test:

The testing was done on a neural network with 2 inputs, one inner layer with 8 neurons and one output layer. The training dataset with 402 inputs was trained for 200 iterations/epochs with a learning rate of 0.1. On testing with the test data with the above specifications, one of the obtained results gave an accuracy of about 87%. The loss vs Epoch graph obtained is given in Figure: 3. The loss values at each epoch is put out in a file log.txt which can be further used is required.

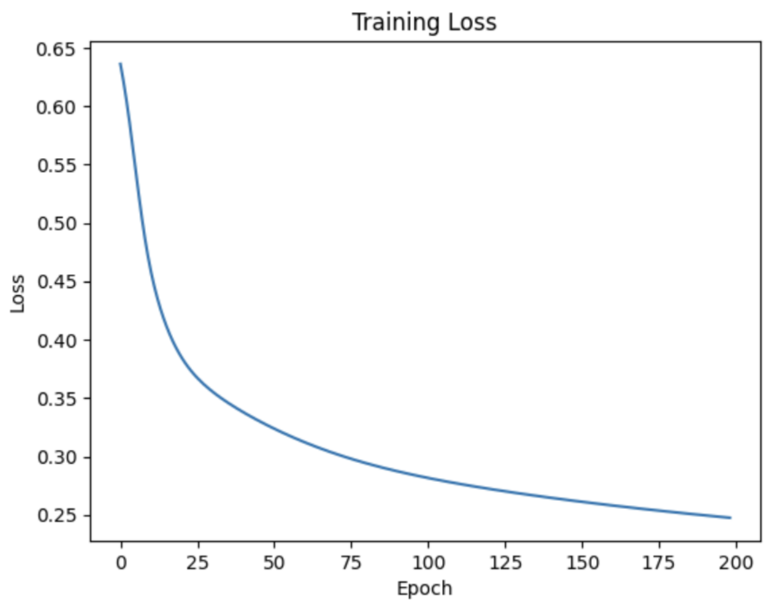


Figure 3: Loss vs Epoch graph

The trained network obtained after this is further tested by creating a fine grid ranging from the smallest At1 value to the largest At1 value and ranging from the smallest At2 value to the largest At2 value in the banana dataset and forming a fine grid between these values having about 75 rows and columns. The intersections of the rows and the columns are taken as inputs for the next test. Upon passing these inputs through the network the predictions are recorded and plotted according to the class it has predicted. The obtained plot is shown in Figure: 4.

To check and compare this with the actual banana dataset we can convert this to a background contour and place the plot of the banana dataset on top of this and compare how well our network can predict and see if the model is underfitting, overfitting or is it a good fit. Figure: 5 shows the same.

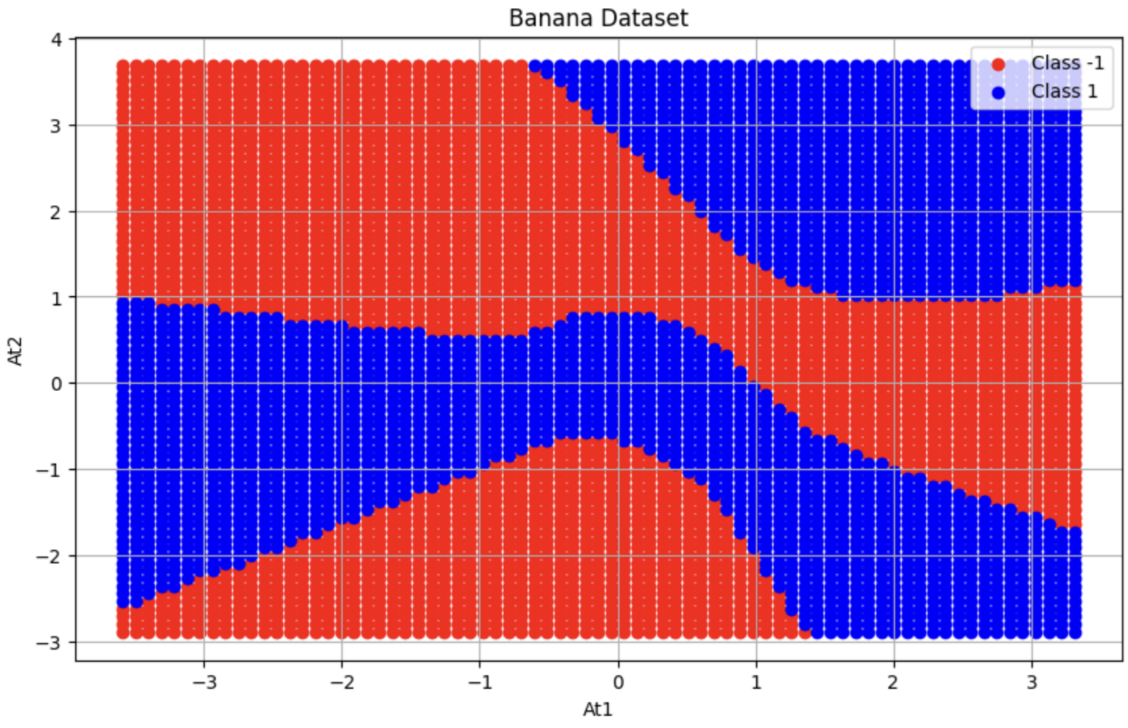


Figure 4: Grid points as inputs

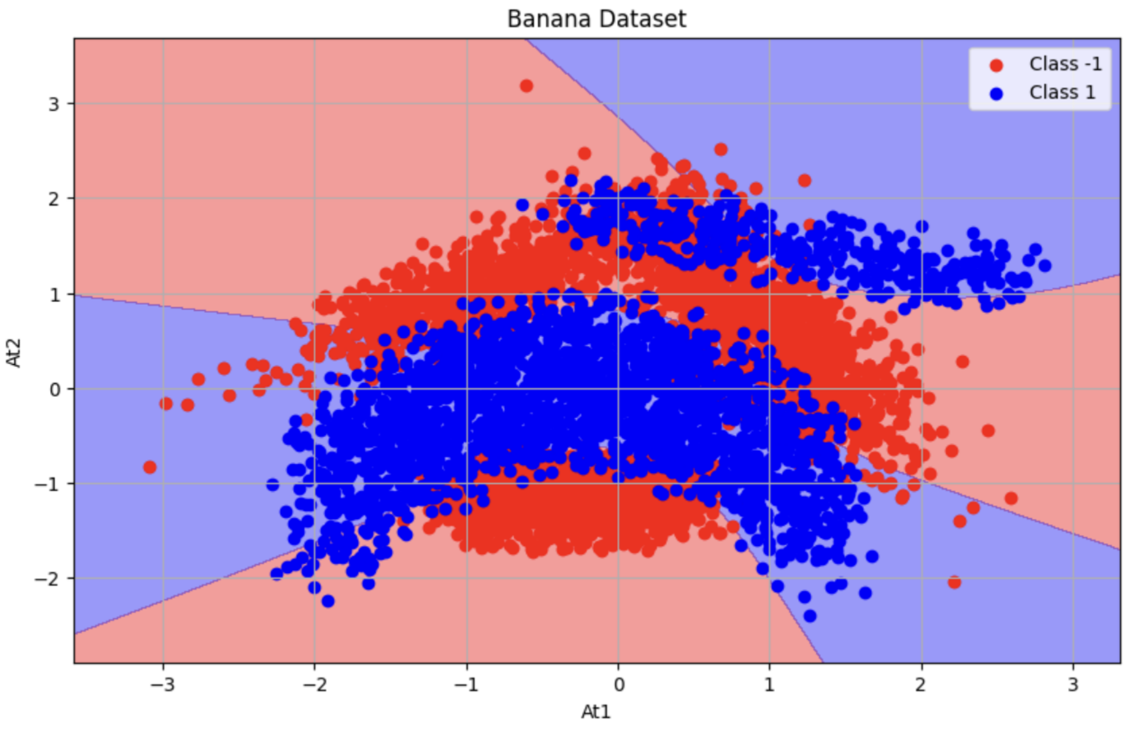


Figure 5: Comparison