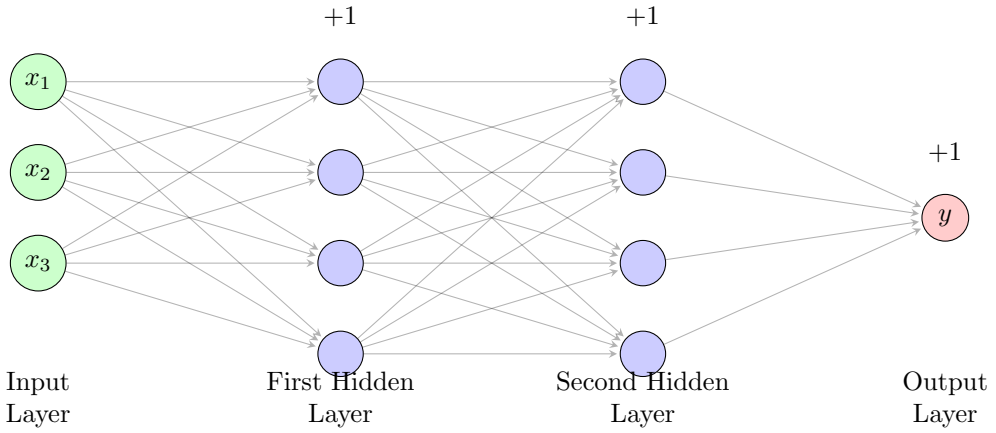
1 Neural Network Parameters Calculation

1.1 Problem Statement

Consider a neural network with:

- 3 input features (x_1, x_2, x_3)
- 2 hidden layers with 4 neurons each
- 1 output neuron
- Bias terms included for each neuron

1.2 Network Architecture



1.3 Parameter Calculation

1.3.1 1. First Hidden Layer

- Weights: Each of the 4 neurons receives connections from all 3 inputs
- Number of weights = $3 \times 4 = 12$
- Bias terms: One for each neuron = 4
- Total parameters for first layer = $(3 \times 4) + 4 = 16$

1.3.2 2. Second Hidden Layer

- Weights: Each of the 4 neurons receives connections from all 4 neurons in previous layer
- Number of weights = $4 \times 4 = 16$
- Bias terms: One for each neuron = 4
- Total parameters for second layer = $(4 \times 4) + 4 = 20$

1.3.3 3. Output Layer

- Weights: The output neuron receives connections from all 4 neurons in previous layer
- Number of weights = $4 \times 1 = 4$
- Bias terms: One for the output neuron = 1
- Total parameters for output layer = $(4 \times 1) + 1 = 5$

1.3.4 Total Parameters

$$\begin{aligned}\text{Total} &= \text{First Hidden Layer} + \text{Second Hidden Layer} + \text{Output Layer} \\ &= (3 \times 4 + 4) + (4 \times 4 + 4) + (4 \times 1 + 1) \\ &= 16 + 20 + 5 \\ &= 41 \text{ parameters}\end{aligned}$$

2 Gradient Boosted Regression Tree (GBRT) Calculation

2.1 Problem Statement

Given the ground truth (y) and predictions ($\hat{y}_1$) from the first decision tree in a GBRT, we need to:

1. Calculate the target values for the second tree
2. Determine the final predictions given that points 1,2,3 are in the left node and 4,5 are in the right node of the second tree

2.2 Initial Data and Calculations

Data Point	y	$\hat{y}_1$	Target for 2^{nd}	Final Prediction
1	250	220		
2	300	280		
3	200	230		
4	350	310		
5	280	300		

Table 1: GBRT Predictions and Targets

2.3 Step-by-Step Solution

2.3.1 1. Calculating Targets for Second Tree

The target for the second tree is the residual (error) from the first tree:

$$\text{Target} = y - \hat{y}_1$$

For each point:

- Point 1: $250 - 220 = 30$
- Point 2: $300 - 280 = 20$
- Point 3: $200 - 230 = -30$
- Point 4: $350 - 310 = 40$
- Point 5: $280 - 300 = -20$

2.3.2 2. Second Tree Predictions

Given the split (1,2,3 in left node; 4,5 in right node):

Left Node Average:

$$\frac{30 + 20 + (-30)}{3} = 6.67$$

Right Node Average:

$$\frac{40 + (-20)}{2} = 10$$

2.3.3 3. Final Predictions

Final prediction = First tree prediction + Second tree prediction

For left node points (1,2,3):

- Point 1: $220 + 6.67 = 226.67$
- Point 2: $280 + 6.67 = 286.67$
- Point 3: $230 + 6.67 = 236.67$

For right node points (4,5):

- Point 4: $310 + 10 = 320.00$
- Point 5: $300 + 10 = 310.00$

2.4 Key Points

- The second tree learns the residuals from the first tree
- Node predictions are the average of the residuals in that node
- Final predictions combine both trees' outputs
- The splitting of nodes helps reduce the overall prediction error

Data Point	y	$\hat{y}_1$	Target for 2^{nd}	Final Prediction
1	250	220	30	226.67
2	300	280	20	286.67
3	200	230	-30	236.67
4	350	310	40	320.00
5	280	300	-20	310.00

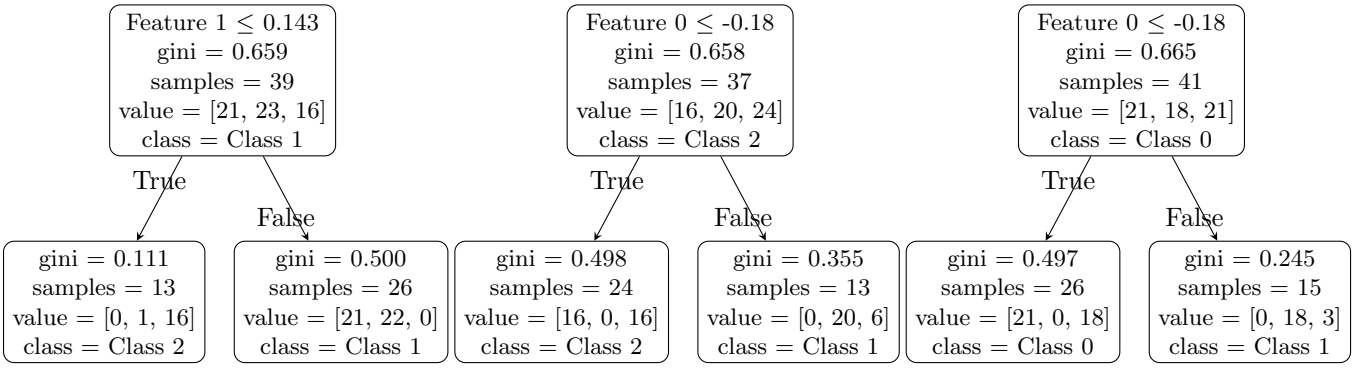
Table 2: GBRT Predictions and Targets

3 Feature Importance in Decision Stumps

3.1 Problem Statement

Calculate the feature importance of Feature 0 and Feature 1 for three decision stumps (decision trees with max_depth=1).

3.2 Tree Structures



3.3 Feature Importance Calculation

Feature importance is calculated based on the reduction in Gini impurity weighted by the number of samples. For each tree:

$$\text{Importance} = \text{parent\_gini} - \frac{n_{\text{left}}}{n_{\text{total}}} \times \text{left\_gini} - \frac{n_{\text{right}}}{n_{\text{total}}} \times \text{right\_gini}$$

3.3.1 Tree 1 (Feature 1)

$$\begin{aligned}\text{Importance} &= 0.659 - \frac{13}{39} \times 0.111 - \frac{26}{39} \times 0.500 \\ &= 0.659 - 0.037 - 0.333 \\ &= 0.289\end{aligned}$$

3.3.2 Tree 2 (Feature 0)

$$\begin{aligned}\text{Importance} &= 0.658 - \frac{24}{37} \times 0.498 - \frac{13}{37} \times 0.355 \\ &= 0.658 - 0.323 - 0.125 \\ &= 0.210\end{aligned}$$

3.3.3 Tree 3 (Feature 0)

$$\begin{aligned}\text{Importance} &= 0.665 - \frac{26}{41} \times 0.497 - \frac{15}{41} \times 0.245 \\ &= 0.665 - 0.315 - 0.090 \\ &= 0.260\end{aligned}$$

3.4 Final Feature Importance

- **Feature 0**: Average of Trees 2 and 3 = $\frac{0.210+0.260}{0.210+0.260+0.289} \times 100\% = 61.9\%$
- **Feature 1**: From Tree 1 = $\frac{0.289}{0.210+0.260+0.289} \times 100\% = 38.1\%$

4 Attention Mechanism

Consider a sequence $\mathbf{x1}, \mathbf{x2}$ of two embedded token vectors in 2 dimensions:

$$\mathbf{x1} = \begin{bmatrix} 0.1 & 0.4 \end{bmatrix}^\top, \quad \mathbf{x2} = \begin{bmatrix} 0.5 & 0.3 \end{bmatrix}^\top.$$

Consider one head with $\mathbf{W^Q} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$, $\mathbf{W^K} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$, and $\mathbf{W^V} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. a) Calculate the attention matrix. Show the steps. Use scaling factor as 1. b) Calculate the context-aware **z1** and **z2** representations.

4.1 Attention Calculation

$$q_1 = \mathbf{W^Q x1} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 0.6 \end{bmatrix}$$
$$q_2 = \mathbf{W^Q x2} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 1.1 \\ 1.3 \end{bmatrix}$$
$$k_1 = \mathbf{W^K x1} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix}$$
$$k_2 = \mathbf{W^K x2} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.8 \end{bmatrix}$$
$$v_1 = \mathbf{W^V x1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.4 \end{bmatrix}$$
$$v_2 = \mathbf{W^V x2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix}$$

The attention scores are calculated as:

$$q_1 \cdot k_1 = \begin{bmatrix} 0.9 \\ 0.6 \end{bmatrix} \cdot \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix} = 0.9 \times 0.4 + 0.6 \times 0.5 = 0.36 + 0.3 = 0.66$$
$$q_1 \cdot k_2 = \begin{bmatrix} 0.9 \\ 0.6 \end{bmatrix} \cdot \begin{bmatrix} 0.3 \\ 0.8 \end{bmatrix} = 0.9 \times 0.3 + 0.6 \times 0.8 = 0.27 + 0.48 = 0.75$$
$$q_2 \cdot k_1 = \begin{bmatrix} 1.1 \\ 1.3 \end{bmatrix} \cdot \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix} = 1.1 \times 0.4 + 1.3 \times 0.5 = 0.44 + 0.65 = 1.09$$
$$q_2 \cdot k_2 = \begin{bmatrix} 1.1 \\ 1.3 \end{bmatrix} \cdot \begin{bmatrix} 0.3 \\ 0.8 \end{bmatrix} = 1.1 \times 0.3 + 1.3 \times 0.8 = 0.33 + 1.04 = 1.37$$

The attention matrix is calculated as:

$$\begin{bmatrix} 0.66 & 0.75 \\ 1.09 & 1.37 \end{bmatrix} \xrightarrow{Softmax} \begin{bmatrix} 0.478 & 0.522 \\ 0.43 & 0.57 \end{bmatrix}$$

The context-aware representations are calculated as:

$$\mathbf{z1} = 0.478 \times \begin{bmatrix} 0.1 \\ 0.4 \end{bmatrix} + 0.522 \times \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0.478 \times 0.1 + 0.522 \times 0.5 \\ 0.478 \times 0.4 + 0.522 \times 0.3 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.34 \end{bmatrix}$$
$$\mathbf{z2} = 0.43 \times \begin{bmatrix} 0.1 \\ 0.4 \end{bmatrix} + 0.57 \times \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0.43 \times 0.1 + 0.57 \times 0.5 \\ 0.43 \times 0.4 + 0.57 \times 0.3 \end{bmatrix} = \begin{bmatrix} 0.32 \\ 0.34 \end{bmatrix}$$

5 CNN

Consider a convolutional network with the following layers – Input (100x100x3), Conv1 (100x100x32), MaxPool1 (50x50x32), Conv2 (25x25xA), Flatten, Dense1 (100), Output (5). Each Conv2D Layer uses a 3x3 kernel and MaxPoolingLayer uses a 2x2 pool. This network is used to perform a 5-class image classification problem.

5.1 Question 1

What is the configuration used in Conv2 to get 25x25? Is it possible to get 25x25?

5.2 Answer

Stride: The stride used in the Conv2 layer is 2. This means that the convolution kernel moves 2 steps horizontally and vertically, effectively downsampling the feature maps.

Feasibility: Yes, it is possible to get a 25x25 output feature map size from the Conv2 layer. The input to the Conv2 layer is 100x100x32, and with a 3x3 kernel size and a stride of 2, the output size can be calculated as:

$$\text{Output Size} = \left\lfloor \frac{100 - 3 + 1}{2} \right\rfloor = 25$$

Therefore, the output of the Conv2 layer will have a spatial dimension of 25x25.

5.3 Question 2

Does the image size (spatial dimensions) affect the number of parameters that need to be learnt in this CNN? Why or why not?

5.4 Answer

Yes, the image size (spatial dimensions) affects the number of parameters that need to be learnt in this CNN. This is because the number of parameters in the convolutional layers depends on the spatial dimensions of the input and the number of channels.

Specifically, in a convolutional layer, the number of parameters is determined by the kernel size, the number of input channels, and the number of output channels. The spatial dimensions of the input affect the number of times the convolutional kernel is applied, which in turn affects the total number of parameters.

For example, if the input has a larger spatial size, the convolutional layer will have more spatial locations to apply the kernel, leading to more parameters to be learned. Conversely, if the input has a smaller spatial size, the convolutional layer will have fewer spatial locations, resulting in fewer parameters to be learned.

Therefore, the image size (spatial dimensions) is an important factor that affects the complexity and number of parameters in the CNN architecture.

5.5 Question 3

Find the value of A if a total of 315446 parameters have to be learnt.

5.6 Answer

To find the value of A, we need to calculate the total number of parameters in the given convolutional network.

The total number of parameters can be expressed as:

$$\text{Total Parameters} = 28 \times 32(Conv1) + 289A(Conv2) + 65200A + 100 + 505(\text{Dense Layer})$$

Equating the total parameters to the given value of 315446, we can solve for the value of A:

$$A = 5$$

Therefore, the value of A is 5.

5.7 Question 4

How many total parameters need to be learnt if Conv1 is replaced with DepthWise Separable Layer? (If you can't find A in Q.4, leave your answer in terms of A)

5.8 Answer

If Conv1 is replaced with a Depthwise Separable Layer, the number of total parameters that need to be learnt will be reduced.

In a standard convolutional layer, the number of parameters is given by:

$$\text{Parameters} = K \times K \times C_{\text{in}} \times C_{\text{out}}$$

where K is the kernel size, C_{in} is the number of input channels, and C_{out} is the number of output channels.

In a Depthwise Separable Layer, the number of parameters is reduced to:

$$\text{Parameters} = K \times K \times C_{\text{in}} + C_{\text{in}} \times C_{\text{out}}$$

Applying this to the given network, the number of parameters for Conv1 (100x100x3) would be reduced from 896 to $3 \times 3 \times 3 + 3 \times 32 = 155$.

Therefore, the total number of parameters that need to be learnt if Conv1 is replaced with a Depthwise Separable Layer is:

$$\begin{aligned} \text{Total Parameters} &= 315446 - 896 + 155 \\ &= 314705 \end{aligned}$$