MLP Calculations Example

May 13, 2024

ⓘThis conversation may reflect the link creator’s personalized data, which isn’t shared and can meaningfully change how the model responds.

I'm working to solve a reverse engineering challenge about an MLP with a single hidden layer. Can you explain what an MLP is in detail? Walk me through the calculations for a small MLP with a few input bits.

Sure! Let's go through what a Multilayer Perceptron (MLP) is and then walk through the calculations for a small example.

What is a Multilayer Perceptron (MLP)?

An MLP is a type of feedforward artificial neural network that consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Each layer is fully connected to the next one. The nodes in the input layer represent the input features, while the nodes in the hidden and output layers are artificial neurons with an activation function.

Components of an MLP

1. **Input Layer**: Takes the input features.
2. **Hidden Layer**: Consists of neurons that apply a transformation to the inputs using weights, biases, and activation functions.
3. **Output Layer**: Produces the final output of the network.

Activation Function

An activation function introduces non-linearity into the model, allowing it to learn complex patterns. Common activation functions include:

- Sigmoid
- Tanh
- ReLU (Rectified Linear Unit)

Example Calculation

Let's consider a simple MLP with:

- 2 input nodes
- 2 hidden nodes
- 1 output node
- Sigmoid activation function

Weights and Biases

Let’s define the weights and biases for our MLP:

- Weights between input and hidden layer: $W_{ih} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$

- Biases for hidden layer: $b_h = [\begin{smallmatrix} b_{h1} \\ b_{h2} \end{smallmatrix}]$
- Weights between hidden and output layer: $W_{ho} = [\begin{smallmatrix} w_{h1} \\ w_{h2} \end{smallmatrix}]$
- Bias for output layer: b_o

Input

Let's assume our input vector $X = [\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}]$.

Calculations

1. **Hidden Layer Calculations:**

   The input to the hidden layer Z_h is calculated as:

   $$Z_h = W_{ih} \cdot X + b_h$$

   Let's compute this step-by-step:

   $$Z_{h1} = w_{11}x_1 + w_{21}x_2 + b_{h1}$$

   $$Z_{h2} = w_{12}x_1 + w_{22}x_2 + b_{h2}$$

   Apply the sigmoid activation function $\sigma(z) = \frac{1}{1+e^{-z}}$:

   $$A_{h1} = \sigma(Z_{h1}) = \frac{1}{1 + e^{-Z_{h1}}}$$

   $$A_{h2} = \sigma(Z_{h2}) = \frac{1}{1 + e^{-Z_{h2}}}$$

2. **Output Layer Calculations:**

   The input to the output layer Z_o is calculated as:

   $$Z_o = W_{ho} \cdot A_h + b_o$$

   Let's compute this step-by-step:

   $$Z_o = w_{h1}A_{h1} + w_{h2}A_{h2} + b_o$$

   Apply the sigmoid activation function to get the final output A_o:

   $$A_o = \sigma(Z_o) = \frac{1}{1 + e^{-Z_o}}$$

Example with Specific Values

Let's use some specific values:

- Input: $X = \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix}$
- Weights: $W_{ih} = \begin{bmatrix} 0.2 & 0.4 \\ 0.6 & 0.8 \end{bmatrix}$
- Biases for hidden layer: $b_h = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$
- Weights for output layer: $W_{ho} = \begin{bmatrix} 0.5 \\ 0.7 \end{bmatrix}$
- Bias for output layer: $b_o = 0.1$

Hidden Layer Calculations

$$Z_{h1} = 0.2 \cdot 0.5 + 0.6 \cdot 0.3 + 0.1 = 0.35$$

$$Z_{h2} = 0.4 \cdot 0.5 + 0.8 \cdot 0.3 + 0.1 = 0.55$$

Apply the sigmoid activation function:

$$A_{h1} = \sigma(0.35) = \frac{1}{1 + e^{-0.35}} \approx 0.586$$

$$A_{h2} = \sigma(0.55) = \frac{1}{1 + e^{-0.55}} \approx 0.634$$

Output Layer Calculations

$$Z_o = 0.5 \cdot 0.586 + 0.7 \cdot 0.634 + 0.1 \approx 0.827$$

Apply the sigmoid activation function:

$$A_o = \sigma(0.827) = \frac{1}{1 + e^{-0.827}} \approx 0.695$$

So, the final output of this MLP given the input $\begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix}$ is approximately 0.695.

This step-by-step process illustrates how an MLP computes its output from the input through a series of matrix multiplications, additions, and the application of activation functions.

What if the output layer has the same number of bits as the input layer? Can you show me an example like that?

Sure! Let's consider an MLP where the number of bits in the output layer is the same as in the input layer. We'll follow a similar approach but with more nodes in the output layer.

MLP with Same Number of Input and Output Nodes

Let's assume:

- 2 input nodes
- 2 hidden nodes
- 2 output nodes

- Sigmoid activation function

Weights and Biases

Let’s define the weights and biases for our MLP:

- Weights between input and hidden layer: $W_{ih} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$
- Biases for hidden layer: $b_h = \begin{bmatrix} b_{h1} \\ b_{h2} \end{bmatrix}$
- Weights between hidden and output layer: $W_{ho} = \begin{bmatrix} w_{h1} & w_{h2} \\ w_{h3} & w_{h4} \end{bmatrix}$
- Biases for output layer: $b_o = \begin{bmatrix} b_{o1} \\ b_{o2} \end{bmatrix}$

Input

Let’s assume our input vector $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

Calculations

1. **Hidden Layer Calculations:**

   The input to the hidden layer Z_h is calculated as:

   $$Z_h = W_{ih} \cdot X + b_h$$

   Let's compute this step-by-step:

   $$Z_{h1} = w_{11}x_1 + w_{21}x_2 + b_{h1}$$

   $$Z_{h2} = w_{12}x_1 + w_{22}x_2 + b_{h2}$$

   Apply the sigmoid activation function $\sigma(z) = \frac{1}{1+e^{-z}}$:

   $$A_{h1} = \sigma(Z_{h1}) = \frac{1}{1 + e^{-Z_{h1}}}$$

   $$A_{h2} = \sigma(Z_{h2}) = \frac{1}{1 + e^{-Z_{h2}}}$$

2. **Output Layer Calculations:**

   The input to the output layer Z_o is calculated as:

   $$Z_o = W_{ho} \cdot A_h + b_o$$

   Let's compute this step-by-step:

   $$Z_{o1} = w_{h1}A_{h1} + w_{h3}A_{h2} + b_{o1}$$

$$Z_{o2} = w_{h2}A_{h1} + w_{h4}A_{h2} + b_{o2}$$

Apply the sigmoid activation function to get the final outputs A_{o1} and A_{o2}:

$$A_{o1} = \sigma(Z_{o1}) = \frac{1}{1 + e^{-Z_{o1}}}$$

$$A_{o2} = \sigma(Z_{o2}) = \frac{1}{1 + e^{-Z_{o2}}}$$

Example with Specific Values

Let's use some specific values:

- Input: $X = \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix}$
- Weights: $W_{ih} = \begin{bmatrix} 0.2 & 0.4 \\ 0.6 & 0.8 \end{bmatrix}$
- Biases for hidden layer: $b_h = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$
- Weights for output layer: $W_{ho} = \begin{bmatrix} 0.5 & 0.7 \\ 0.6 & 0.9 \end{bmatrix}$
- Biases for output layer: $b_o = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$

Hidden Layer Calculations

$$Z_{h1} = 0.2 \cdot 0.5 + 0.6 \cdot 0.3 + 0.1 = 0.35$$

$$Z_{h2} = 0.4 \cdot 0.5 + 0.8 \cdot 0.3 + 0.1 = 0.55$$

Apply the sigmoid activation function:

$$A_{h1} = \sigma(0.35) = \frac{1}{1 + e^{-0.35}} \approx 0.586$$

$$A_{h2} = \sigma(0.55) = \frac{1}{1 + e^{-0.55}} \approx 0.634$$

Output Layer Calculations

$$Z_{o1} = 0.5 \cdot 0.586 + 0.6 \cdot 0.634 + 0.1 \approx 0.777$$

$$Z_{o2} = 0.7 \cdot 0.586 + 0.9 \cdot 0.634 + 0.1 \approx 1.075$$

Apply the sigmoid activation function:

$$A_{o1} = \sigma(0.777) = \frac{1}{1 + e^{-0.777}} \approx 0.685$$

$$A_{o2} = \sigma(1.075) = \frac{1}{1 + e^{-1.075}} \approx 0.746$$

So, the final outputs of this MLP given the input $\begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix}$ are approximately 0.685 and 0.746.

Can you explain this as pseudocode instead of matrix multiplications?

Sure! Here's how you can describe the operations of an MLP in pseudocode, focusing on the step-by-step calculations without using matrix operations explicitly.

Pseudocode for MLP

Initialization

Define the weights and biases for the layers:

```plaintext
# Weights between input and hidden layer
weights_ih = [
    [w11, w12],  # Weights for hidden neuron 1
    [w21, w22]   # Weights for hidden neuron 2
]

# Biases for hidden layer
biases_h = [b_h1, b_h2]

# Weights between hidden and output layer
weights_ho = [
    [w_h1, w_h3],  # Weights for output neuron 1
    [w_h2, w_h4]   # Weights for output neuron 2
]

# Biases for output layer
biases_o = [b_o1, b_o2]

# Sigmoid activation function
function sigmoid(x):
    return 1 / (1 + exp(-x))
```

Forward Pass

Given an input $X = [x1, x2]$:

1. **Hidden Layer Calculations:**

```plaintext
# Calculate the input to hidden neurons
z_h1 = weights_ih[0][0] * x1 + weights_ih[1][0] * x2 + biases_h[0]
z_h2 = weights_ih[0][1] * x1 + weights_ih[1][1] * x2 + biases_h[1]

# Apply the sigmoid activation function
a_h1 = sigmoid(z_h1)
a_h2 = sigmoid(z_h2)
```

2. **Output Layer Calculations:**

```plaintext
# Calculate the input to output neurons
z_o1 = weights_ho[0][0] * a_h1 + weights_ho[1][0] * a_h2 + biases_o[0]
z_o2 = weights_ho[0][1] * a_h1 + weights_ho[1][1] * a_h2 + biases_o[1]
```

```
# Apply the sigmoid activation function
a_o1 = sigmoid(z_o1)
a_o2 = sigmoid(z_o2)
```

Example with Specific Values

Let's use specific values as an example:

- Input: $X = [0.5, 0.3]$
- Weights and biases:

  ```plaintext
  weights_ih = [
      [0.2, 0.4],
      [0.6, 0.8]
  ]
  biases_h = [0.1, 0.1]
  weights_ho = [
      [0.5, 0.7],
      [0.6, 0.9]
  ]
  biases_o = [0.1, 0.1]
  ```

Forward Pass Example

1. **Hidden Layer Calculations:**

```plaintext
# Calculate the input to hidden neurons
z_h1 = 0.2 * 0.5 + 0.6 * 0.3 + 0.1
# z_h1 = 0.1 + 0.18 + 0.1 = 0.35

z_h2 = 0.4 * 0.5 + 0.8 * 0.3 + 0.1
# z_h2 = 0.2 + 0.24 + 0.1 = 0.55

# Apply the sigmoid activation function
a_h1 = sigmoid(0.35)  # ≈ 0.586
a_h2 = sigmoid(0.55)  # ≈ 0.634
```

2. **Output Layer Calculations:**

```plaintext
# Calculate the input to output neurons
z_o1 = 0.5 * 0.586 + 0.6 * 0.634 + 0.1
# z_o1 ≈ 0.293 + 0.380 + 0.1 = 0.773

z_o2 = 0.7 * 0.586 + 0.9 * 0.634 + 0.1
# z_o2 ≈ 0.410 + 0.571 + 0.1 = 1.081

# Apply the sigmoid activation function
a_o1 = sigmoid(0.773)  # ≈ 0.684
a_o2 = sigmoid(1.081)  # ≈ 0.747
```

So, the final outputs given the input [0.5, 0.3] are approximately 0.684 and 0.747. This pseudocode outlines the steps without relying on matrix operations, making it clear how the input values propagate through the network to produce the final output.