Loss Function, Activation Function, and Weight Update using the Chain Rule:

Neural networks use the **chain rule of differentiation** to calculate how the loss changes with respect to weights and biases (backpropagation). Let's break it down step by step:

---

1. Key Components

- **Loss Function (L)**: Measures the difference between the predicted output ($\hat{y}$) and the actual output (y).
  - Example:
    - **Mean Squared Error (MSE)** for regression:
      $L = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$
    - **Cross-Entropy Loss** for classification:
      $L = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$
- **Activation Function ($f(z)$)**: Adds non-linearity to the output of a neuron.
  - $z = W \cdot X + b$, where W is the weight matrix, X is the input, and b is the bias.
  - Examples of $f(z)$:
    - **Sigmoid**: $f(z) = \frac{1}{1+e^{-z}}$
    - **ReLU**: $f(z) = \max(0, z)$
- **Weights (W) and Biases (b)**: Parameters of the model that are updated during training.

---

2. Chain Rule in Backpropagation

To update weights and biases, we calculate:

- **Loss w.r.t weights** ($\frac{\partial L}{\partial W}$)
- **Loss w.r.t biases** ($\frac{\partial L}{\partial b}$)

Using the **chain rule**:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial f(z)} \cdot \frac{\partial f(z)}{\partial z} \cdot \frac{\partial z}{\partial W}$$

Similarly, for biases:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial f(z)} \cdot \frac{\partial f(z)}{\partial z} \cdot \frac{\partial z}{\partial b}$$

---

3. Step-by-Step Example

Problem:

We have:

- **Input** (X): $X = [1, 2]$
- **Weights** (W): $W = [0.5, 0.8]$
- **Bias** (b): $b = 0.2$
- **Activation Function**: Sigmoid ($f(z) = \frac{1}{1+e^{-z}}$)

- **Loss Function**: MSE ($L = (\hat{y} - y)^2$), where $y = 1$ (true label).

Steps:

1. **Forward Pass**:
    - Calculate z:

$$z = W \cdot X + b = (0.5 \cdot 1 + 0.8 \cdot 2) + 0.2 = 2.3$$

    - Apply the activation function ($\hat{y}$):

$$\hat{y} = f(z) = \frac{1}{1 + e^{-2.3}} \approx 0.908$$

    - Compute the loss:

$$L = (\hat{y} - y)^2 = (0.908 - 1)^2 = 0.0084$$

2. **Backward Pass** (Using Chain Rule):
    - **Loss w.r.t predicted output**:

$$\frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y) = 2(0.908 - 1) = -0.184$$

    - **Predicted output w.r.t** z (derivative of sigmoid):

$$\frac{\partial \hat{y}}{\partial z} = \hat{y}(1 - \hat{y}) = 0.908(1 - 0.908) \approx 0.083$$

    - z **w.r.t weights**:

$$\frac{\partial z}{\partial W} = X$$

3. **Calculate Gradients**:

- Gradient w.r.t W:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial W}$$

  Substitute:

$$\frac{\partial L}{\partial W} = (-0.184)(0.083)([1, 2]) \approx [-0.0153, -0.0306]$$

- Gradient w.r.t b:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} = (-0.184)(0.083) \approx -0.0153$$

4. **Update Weights and Bias** (Gradient Descent):

- Learning Rate (η): Assume $\eta = 0.1$.
- Update rule:

$$W_{\text{new}} = W_{\text{old}} - \eta \cdot \frac{\partial L}{\partial W}$$

$$b_{\text{new}} = b_{\text{old}} - \eta \cdot \frac{\partial L}{\partial b}$$

- Updated weights:

$$W_{\text{new}} = [0.5, 0.8] - 0.1 \cdot [-0.0153, -0.0306] \approx [0.5015, 0.803]$$

- Updated bias:

$$b_{\text{new}} = 0.2 - 0.1 \cdot (-0.0153) \approx 0.2015$$

---

4. Summary

- **Forward Pass**: Compute predictions and loss.
- **Backward Pass**: Use chain rule to calculate gradients ($\frac{\partial L}{\partial W}$, $\frac{\partial L}{\partial b}$).
- **Weight Update**: Adjust weights and biases using gradient descent.

This process is repeated for all layers in the network during backpropagation to minimize the loss and train the neural network.