Part-c

Given:

* Model Architecture : 2-Layer Feed forward Neural Network
* Activation Function : Hyperbolic Tangent (tanh) for all hidden and output layer neurons
* Algorithm : Back-propagation for weight updates

Output format (Required)

* Explicit formulas for weight update rules
* Formulas must be in a matrix notation
* All Symbols and their dimensions must be defined
* All gradients used in the update rules must be derived.

Solution:

1) Symbol definitions and dimensions

* n_i : Number of input neurons
* n_h : Number of hidden neurons
* n_o : Number of output neurons
* $x \in \mathbb{R}^{n_i \times 1}$: Input vector
* $W^1 \in \mathbb{R}^{n_h \times n_i}$: Weight matrix for the hidden layer
* $b^1 \in \mathbb{R}^{n_h \times 1}$: Bias vector for the hidden layer
* $z^1 \in \mathbb{R}^{n_h \times 1}$: Net input to the hidden layer, $z^1 = W^1 x + b^1$
* $a^1 \in \mathbb{R}^{n_h \times 1}$: Activation of the hidden layer, $a^1 = \tanh(z^1)$
* $W^2 \in \mathbb{R}^{n_o \times n_h}$: Weight matrix of the output layer
* $b^2 \in \mathbb{R}^{n_o \times 1}$: Bias vector for the output layer
* $z^2 \in \mathbb{R}^{n_o \times 1}$: Net input to the output layer, $z^2 = W^2 a^1 + b^2$
* $a^2 \in \mathbb{R}^{n_o \times 1}$: Activation of the output layer, $a^2 = \tanh(z^2)$
* $y \in \mathbb{R}^{n_o \times 1}$: Target output vector
* $J = \frac{1}{2}\|y - a^2\|^2$: Mean squared error loss function

Derivative of hyperbolic tangent (tanh) function:

$$\frac{d}{dx}\tanh(x) = 1 - \tanh^2(x)$$

2) Gradient derivation:-

Back propagation uses chain rule to compute the gradient of the loss function w.r.t each weight

a) Output layer Gradients:-

Gradient of the loss with respect to the output layer weights, W^2 and biases, b^2.

Gradient with respect to the output layer's net input, z^2 is:

$$\frac{\partial J}{\partial z^2} = \frac{\partial J}{\partial a^2} \cdot \frac{\partial a^2}{\partial z^2}$$

where $\cdot$ denotes element wise multiplication

$$\frac{\partial J}{\partial a^2} = \frac{\partial}{\partial a^2}\left(\frac{1}{2}\|y - a^2\|^2\right) = -(y - a^2) = a^2 - y$$

$$\frac{\partial a^2}{\partial z^2} = \frac{\partial}{\partial z^2}\tanh(z^2) = 1 - \tanh^2(z^2) = 1 - (a^2)^2$$

Combining these, we get the output layer error delta, δ^2:

$$\delta^2 = \frac{\partial J}{\partial z^2} = (a^2 - y) \odot (1 - (a^2)^2)$$

Computing gradients for the weights and biases of the output layer

$$\frac{\partial J}{\partial W^2} = \frac{\partial J}{\partial z^2}\frac{\partial z^2}{\partial W^2} = \delta^2 (a^1)^T$$

$$\frac{\partial J}{\partial b^2} = \frac{\partial J}{\partial z^2}\frac{\partial z^2}{\partial b^2} = \delta^2$$

b) Hidden layer Gradients:

Back propagate errors to the hidden layer

The gradient with respect to the hidden layer's net input z^1 is

$$\frac{\partial J}{\partial z^1} = \frac{\partial J}{\partial a^2}\frac{\partial a^2}{\partial z^2}\frac{\partial z^2}{\partial a^1}\frac{\partial a^1}{\partial z^1}$$

Simplified using output layer delta

$$\frac{\partial J}{\partial z^1} = \left(\frac{\partial z^2}{\partial a^1}\right)^T \frac{\partial J}{\partial z^2} \odot \frac{\partial a^1}{\partial z^1}$$

$$\frac{\partial J}{\partial z^1} = (W^2)^T \delta^2 \odot (1 - (a^1)^2)$$

Hidden layer error delta, δ^1:

$$\delta^1 = (W^2)^T \delta^2 \odot (1 - (a^1)^2)$$

Using this delta, we can find the gradients for the weights and biases of the hidden layer:

$$\frac{\partial J}{\partial W^1} = \frac{\partial J}{\partial z^1}\frac{\partial z^1}{\partial W^1} = \delta^1 x^T$$

$$\frac{\partial J}{\partial b^1} = \frac{\partial J}{\partial z^1}\frac{\partial z^1}{\partial b^1} = \delta^1$$

3) Weight update rules:-

weight update rule for a parameter Θ is $\Theta_{new} = \Theta_{old} - \alpha\frac{\partial J}{\partial \Theta}$

α - learning rate

Using the gradients derived above, the explicit weight update rules in matrix notation are:

* Update for Output Layer Weights (W^2):

$$W^2_{new} = W^2 - \alpha\delta^2 (a^1)^T$$

* Update for Output layer Biases (b^2):

$$b^2_{new} = b^2 - \alpha\delta^2$$

* Update for Hidden layer weights (W^1)

$$W^1_{new} = W^1 - \alpha\delta^1 x^T$$

* Update for Hidden layer Biases (b^1):

$$b^1_{new} = b^1 - \alpha\delta^1$$

These rules will allow the network's weights and biases to be iteratively adjusted to minimize the loss function

—x—