***Learning about the activation functions in the neural networks***

Neural networks, also called artificial neural networks or simply neural networks, are computational models based on the structure and function of the human brain. They consist of interconnected artificial neurons (also called nodes or units) that work together to process and analyze complex data and make predictions or decisions.

A neural network consists of several layers of interconnected neurons, usually divided into three main layers: Input layer, hidden layer(s), and output layer. The input layer receives the output data, the hidden layer(s) perform computations and transformations on the data, and the output layer produces the final result or prediction.

Each neuron in a neural network receives inputs, applies weights to those inputs, computes an activation function, and produces an output. The activation function determines the neuron's output based on the weighted sum of its inputs, adding a nonlinear element to the network's calculations.

During the training process, neural networks learn from labeled examples or data through a technique called backpropagation. Backpropagation involves adjusting the weights of connections between neurons to minimize the difference between predicted and actual results. This iterative process allows the network to improve its predictions over time.

***Implementation***

The best thing about this neural network implementation is that it demonstrates the complete process of creating and training a neural network using Python. It covers the major steps, including forward propagation, backward propagation, and weight updating.

Here are the notable features of this implementation:

* - Activation function: it uses the sigmoid activation function (f(x) = 1 / (1 + exp(-x))), a commonly used function that maps the neuron's output to a value between 0 and 1. It introduces nonlinearity into the network and allows it to learn complex relationships in the data.One Hidden Layer: The neural network has one hidden layer, which adds an intermediate processing step between the input and output layers. Hidden layers enable the network to learn and extract complex features from the input data.
* - Iris dataset: the widely used iris dataset is used for neural network training. The dataset contains samples with three different target classes and is therefore suitable for multi-class classification tasks.
* - Number of epochs and learning rate: the number of training epochs is set to 30, which means that the network iterates over the entire dataset 30 times during training. A learning rate of 0.5 is used to control the number of weight updates during backpropagation.
* - Backpropagation: the backpropagation algorithm is used to update the weights in the neural network. It calculates the gradient of the loss function with respect to each weight so that the network can adjust the weights in a direction that minimizes the total error.
* - Iterative training: weights and biases are updated after processing each training sample. This approach, known as online learning, allows the network to adjust and change its parameters based on each training sample, which can lead to faster convergence and better generalization.

Overall, this implementation provides a clear and concise demonstration of creating and training a neural network with Python. It covers the essential concepts and techniques, making it a valuable learning resource for understanding the fundamentals of neural networks and their training process.

Snippets of the output:-

>epoch=0, lrate=0.500, error=94.018

>epoch=1, lrate=0.500, error=83.337

>epoch=2, lrate=0.500, error=83.336

>epoch=3, lrate=0.500, error=83.334

>epoch=4, lrate=0.500, error=83.331

>epoch=5, lrate=0.500, error=83.327

>epoch=6, lrate=0.500, error=83.321

>epoch=7, lrate=0.500, error=83.311

>epoch=8, lrate=0.500, error=83.289

>epoch=9, lrate=0.500, error=83.211

>epoch=10, lrate=0.500, error=81.367

>epoch=11, lrate=0.500, error=65.563

>epoch=12, lrate=0.500, error=51.947

>epoch=13, lrate=0.500, error=47.144

>epoch=14, lrate=0.500, error=45.057

>epoch=15, lrate=0.500, error=44.249

>epoch=16, lrate=0.500, error=43.578

>epoch=17, lrate=0.500, error=43.293

>epoch=18, lrate=0.500, error=43.110

>epoch=19, lrate=0.500, error=42.889

>epoch=20, lrate=0.500, error=42.761

As we can this from this output that the error is decreasing with the training epochs. As I have trained only for 30 epochs, if we try with more epochs we can achieve much better accuracy and can reduce the error.