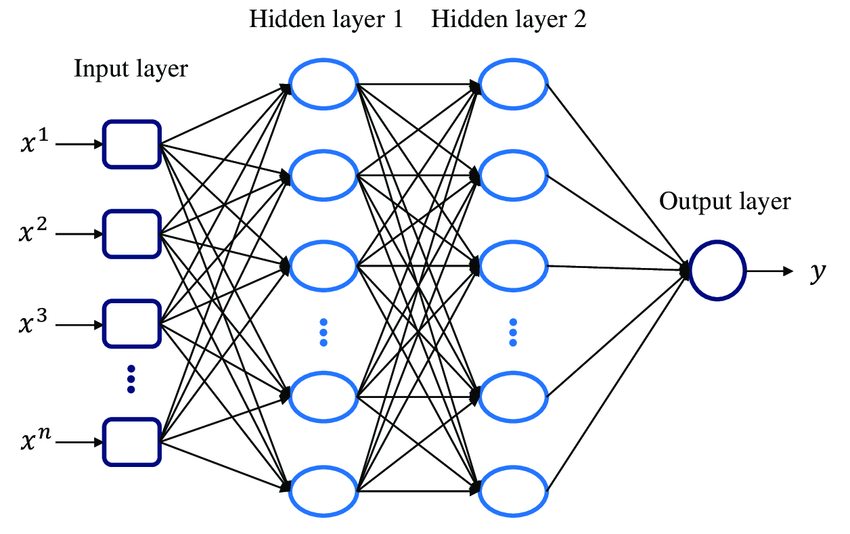
**EXPERIMENT 4**

**Aim: Design, train and test the MLP for tabular data and verify various activation functions and optimizers**

**Description:**



A multi-layer perceptron (MLP) is a class of feed forward artificial neural networks (ANN). An MLP consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer.

**Dataset used:**

The Breast Cancer Wisconsin (Diagnostic) Dataset contains information about various features of breast cell nuclei, which can be used to predict whether a tumor is malignant or benign.

**Key Features of the Dataset:**

* Feature Names:
  + Mean radius, mean texture, mean perimeter, mean area, mean smoothness, mean compactness, mean concavity, mean concave points, mean symmetry, mean fractal dimension, radius error, texture error, perimeter error, area error, smoothness error, compactness error, concavity error, concave points error, symmetry error, fractal dimension error, worst radius, worst texture, worst perimeter, worst area, worst smoothness, worst compactness, worst concavity, worst concave points, worst symmetry, worst fractal dimension.
* **Target Variable:**

Diagnosis: M (malignant), B (benign)

**Common Use Cases:**

* **Binary Classification:** Predicting whether a tumor is malignant or benign based on the given features.

**Architecture Description**

The Breast Cancer dataset is loaded using Scikit-learn's load\_breast\_cancer function.

The create\_model function creates a neural network model with three dense layers (two with dropout). It accepts an activation function and optimizer as parameters and compiles the model with binary cross-entropy loss.

Lists of activation functions (relu, sigmoid, tanh) and optimizers (SGD, Adam, RMSprop) are defined for creating and training multiple models.

Models are trained with different combinations of activation functions and optimizers. Each model is trained on the training data and evaluated on the testing data, with test loss and accuracy printed for comparison.

After training, the results (test loss and accuracy) for each combination of activation function and optimizer are printed to identify the best-performing models.

**Input Layer:**

* 1. Receives 30 input features from the breast cancer dataset.

**Hidden Layer 1:**

* 1. Contains 64 neurons.
  2. Uses the specified activation function (ReLU, sigmoid, or tanh).
  3. Receives input from all 30 input features.
  4. Applies a dropout layer with a 50% dropout rate to prevent overfitting.

**Hidden Layer 2:**

* 1. Contains 32 neurons.
  2. Uses the same activation function as the first hidden layer.
  3. Receives input from all 64 neurons of the previous layer.
  4. Applies another dropout layer with a 50% dropout rate.

**Output Layer:**

* 1. Contains 1 neuron.
  2. Uses a sigmoid activation function to output a probability between 0 and 1.
  3. Represents the probability of the input being a malignant tumor.

**Training Process:**

**Forward Propagation:**

* 1. Input data is fed into the input layer.
  2. The weighted sum of inputs and biases is calculated for each neuron in the hidden layers.
  3. The activation function is applied to the weighted sum, producing the output of the neuron.
  4. This process continues through the hidden layers until the output layer is reached.
  5. The output layer produces the final prediction (probability of malignancy).

**Backpropagation:**

* 1. The error between the predicted output and the actual target is calculated.
  2. The error is propagated backward through the network, adjusting the weights and biases of each neuron using the chosen optimizer (SGD, Adam, or RMSprop).
  3. This process is repeated iteratively until the network learns to make accurate predictions.

**Result Analysis:**

The models were trained using different combinations of activation functions (relu, sigmoid, tanh) and optimizers (SGD, Adam, RMSprop). Here are the key observations:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Activation Function | Optimizer | Test Loss | Test Accuracy | Lr=0.01 | Lr=0.001 |
| Relu | SGD |  |  |  |  |
| Relu | Adam |  |  |  |  |
| Relu | RMSprop |  |  |  |  |
| Sigmoid | SGD |  |  |  |  |
| Sigmoid | Adam |  |  |  |  |
| Sigmoid | RMSprop |  |  |  |  |
| Tanh | SGD |  |  |  |  |
| Tanh | Adam |  |  |  |  |
| Tanh | RMSprop |  |  |  |  |

**From the results, we can conclude that:**

**Model Performance:**

* **Activation Functions:** ?
* **Optimizers:** ?
* **Hyperparameter Tuning:** The learning rate, batch size, and number of epochs = ?

**2. Model Architecture:**

* **Number of Layers and Neurons: *? how many layers can prevent overfitting?***
* **Regularization:** **Dropout= ?**

**Test Accuracy: ?**