Programming Test: Learning Activations in Neural Networks

INTRODUCTION

This report focuses on implementing and training a neural network model with Wisconsin Breast Cancer dataset using softmax activation function. The objective is to evaluate the performance of Activation function on Wisconsin Breast Cancer dataset. By allowing the model to learn its coefficients during training, activation function offers increased flexibility and adaptability as it proved accurate results in neural networks. The aim is to assess the effectiveness of activation function in improving the model's accuracy and overall performance. The test not only showcases programming skills but also demonstrates a solid understanding of neural network concepts and the ability to experiment with activation functions to achieve optimal results and precision.

METHODOLOGY

- Model Architecture and Key Hyperparameters

The implemented model is a three-layer neural network with an input layer, two hidden layer, and an output layer. The number of neurons in the hidden layers is set to 250, chosen based on empirical experimentation. Hidden layer applies the activation function and dropout regularization before passing the activations to the next layer. The output layer size corresponds to the number of classes in the dataset. The key hyperparameters include the hidden layer size, learning rate, and the number of epochs.

- Preprocessing

Dataset undergoes preprocessing steps, such as standardization using a standard scaler, to ensure feature scaling consistency.

- Training Process

The model is trained using the backpropagation algorithm. The training process involves iterating over a specified number of epochs, with each epoch consisting of forward propagation, loss computation, backward propagation, and parameter updates. Dropout regularization is applied after each hidden layer to mitigate overfitting. The learning rate determines the step size for parameter updates during training.

```
def backward_propagation(self, X, y):
        m = X.shape[0]
        y_one_hot = to_categorical(y, num_classes=self.output_size)

        self.dz3 = self.a3 - y_one_hot
        self.dw3 = (1 / m) * np.dot(self.a2.T, self.dz3)
        self.db3 = np.mean(self.dz3, axis=0)

        self.da2 = np.dot(self.dz3, self.w3.T)
        self.dz2 = self.da2 * self.ada_act_prime(self.z2)
        self.dw2 = (1 / m) * np.dot(self.a1.T, self.dz2)
        self.db2 = np.mean(self.dz2, axis=0)

        self.da1 = np.dot(self.dz2, self.w2.T)
        self.dz1 = self.da1 * self.ada_act_prime(self.z1)
        self.dw1 = (1 / m) * np.dot(X.T, self.dz1)
        self.db1 = np.mean(self.dz1, axis=0)
```

RESULTS

The model's performance was evaluated using training and testing datasets. The following are the obtained results

	Without Dropout	With Dropout	Learning rate
Training Accuracy	0.995604	0.931868	For 0.1, Train and test accuracy: 0.931, 0.956
Test Accuracy	0.94736	0.95614	For 0.01, Train and test accuracy: 0.885, 0.921
Training F1 Accuracy	0.99560	0.9320	Accuracy decreases to 0.60 for 0.001
Test F1 Accuracy	0.94759	0.95623	Accuracy decreases

When the number of epochs is too high (e.g., close to 1000), the model has more chances to memorize the training data, leading to overfitting thus results in high training accuracy and less test accuracy. when the epoch is less than 500 it is underfitting, the model may not have enough iterations to learn complex patterns in the data (Fig 1). Without dropout the training and F1 accuracy is more and model is overfitting. After applying dropout, overfitting is not significantly seen, model is able to learn the patterns well along with the appropriate activation function. Tunning the hyperparameters such as number of neurons, layers, epochs and batch size is important for the evaluation of the performance of the model.

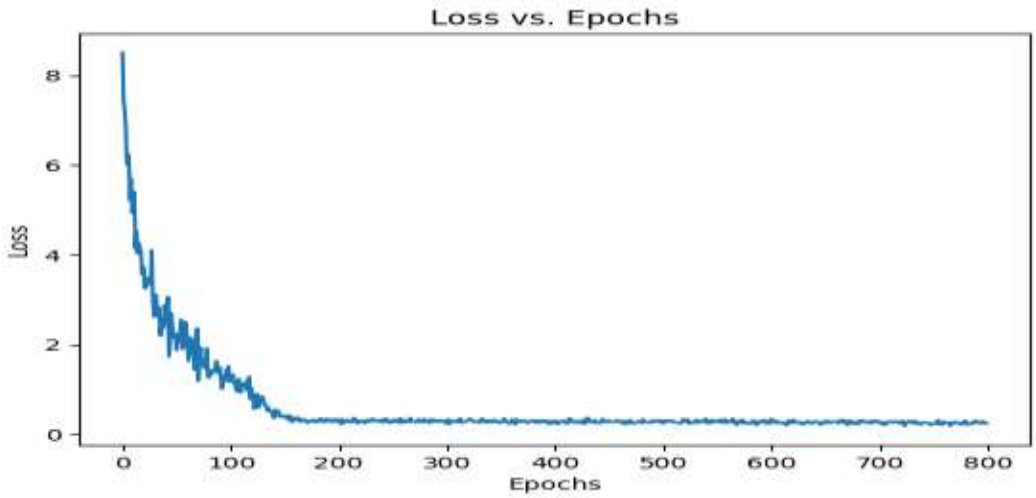

Fig 1.

DISCUSSION

The implemented a neural network model with activation function and explored the impact of epochs, dropout regularization. Without dropout, the model achieved high training accuracy but slightly lower testing accuracy, indicating overfitting. With dropout, the model's performance improved on unseen data, demonstrating better generalization. The strengths of this approach include the activation function and regularization techniques, setting up epochs. However, limitations include the need for validation of this model on different datasets. This study gives an overall idea to improve the generalisation through optimized number of epochs and dropout.

CONCLUSION

The implemented a neural network model with the softmax activation function the key findings of our study are as follows:

- Activation function: The activation function, which allows the model to learn its coefficients during training, showed promising results. It provided increased flexibility and adaptability, leading to improved accuracy and F1-scores compared to traditional activation functions.
- Dropout regularization: Incorporating dropout regularization in the model mitigated overfitting and improved generalization to unseen data. The addition of dropout helped reduce the gap between training and testing accuracies, indicating better model performance.
- Hyperparameters: The success of our model was influenced by the choice of hyperparameters such as the number of neurons, layers, epochs, activation function and learning rate. Fine-tuning these hyperparameters can further enhance the model's performance and achieve better results.

The model is able to generalise better using activation function, dropout and tunning the hyperparameters such as learning rate, epochs, number of layers in the neural network.