Deep Learning Report

Name:	**Pandey Satyam Dhiraj**
Registration No./Roll No.:	21198
Institute/University Name:	IISER Bhopal
Program/Stream:	Physics
Date:	26 March, 2024

Question 1

Logistic regression is used for binary classification through finding a decision boundary. In logistic regression,the output is in range[0,1] which gives the probability of how well the model has matched the actual class labels.In this way it gives a quantitative performance of the linear regression model. MSE penalizes wrong answers much less than CE which leads to higher convergence in case of CE during training. Cross-entropy loss thus provides a way to train models faster towards right direction and get better generalization performance. Let's consider an example to illustrate this:

Suppose we have a binary classification problem where the true label is $y = 1$, and the model's predicted probability is $p(y) = 0.2$. Using both loss functions:

- **Cross Entropy Loss**:

$$\text{Loss} = -(y \log(p(y)) + (1 - y) \log(1 - p(y)))$$

$$\text{Loss} = -(1 \log(0.2) + (1 - 1) \log(1 - 0.2)) = -\log(0.2) \approx 1.609$$

- **Mean Squared Error (MSE)**:

$$\text{Loss} = (y - p(y))^2$$

$$\text{Loss} = (1 - 0.2)^2 = 0.64$$

From the above example, you can see that the Cross Entropy Loss penalizes the model much more for the incorrect prediction compared to MSE. This heavy penalty encourages the model to adjust its weights more aggressively during training to minimize the loss, leading to faster and more effective learning.

Question 2

For a given input X, if all the activation functions in the MLP are linear, then the entire network becomes a linear function of the input X, i.e.

$$\hat{y_i} = w_i X + b_i$$

Now, the loss functions for cross entropy and MSE would be:

$$L_{CE} = -\sum_{i=1}^{2} y_i log(\hat{y_i})$$

$$L_{MSE} = \frac{1}{2}\sum_{i=1}^{2}(y_i - \hat{y}_i)^2$$

If we take the value of y_i=1 when it is the index of the correct class, then we have our function defined as the $f(x) = -log(x)$ We need to compute the double derivative of the function to prove whether the function is convex or not:

$$\frac{\partial L}{\partial x} = -\frac{1}{x} \Rightarrow \frac{\partial^2 L}{\partial x^2} = \frac{1}{x^2} > 0$$

For the MSE loss, too we need to calculate the gradient of the function:

$$\nabla^2 \text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\nabla^2(y_i - \hat{y}_i)^2$$

Since $(y_i - \hat{y}_i)^2$ is a convex function of $\hat{y}_i$, and the sum of convex functions is convex, the MSE loss function is convex.
The answer is option (c) both.

Question 3

1.Architecture:

The feedforward neural network model takes the input size, a list of hidden layer sizes, output size, and a list of activation functions as arguments. It creates a sequential model with linear layers and the specified activation functions.

Let X represent the input data, W represent the weight matrices, b represent the bias vectors, and f represent the activation function. The output of each layer can be calculated as follows:

$I^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$

$O^{[l]} = f(Z^{[l]})$

where $Z^{[l]}$ is the input to layer l, $A^{[l]}$ is the output of layer l, and l denotes the layer index.

2.Preprocessing: 1.Normalization of pixel values to a specific range[0, 1].
2.Flatenning the imgaes for the feed forward network.
3.One-hot encoding the labels

Hyperparameter tuning strategies include grid search, random search, and more advanced techniques like Bayesian optimization. These methods aim to find the optimal set of hyperparameters that maximize the neural network's performance. Techniques like learning rate schedules, weight decay, dropout, and Batch normalization can be employed to improve performance and generalize the model. Here we have used Grid search for evaluation.

Question 4

The training was done on Google Colab,and the entire dataset was used.The training was done for 10 epochs. The training dataset was split into train and validation following a 75-25 ratio. The following results were obtained on each model:

- LeNet :Validation accuracy of 96.4.Test Accuracy is 83.57 and loss is 0.947
- AlexNet:Validation accuracy of 54.Test Accuracy is 51
- VGG11:Loss: 0.7828, Accuracy: 0.7632

The observed performance differences may be attributed to how well the models adapt to the relatively smaller dataset of SVHN. All models, except LeNet-5, were pre-trained on the extensive ImageNet database. When using a significantly smaller SVHN dataset, there is a propensity for underfitting due to the limited amount of data available. Another reason LeNet-5 performs well could be because it is a compact model trained from scratch, enabling it to adapt more effectively to the SVHN dataset.