# Deep Learning Assignement

**Aditya Mishra - 21013**

**March 26, 2024**

The codes for the questions can be accessed in the following Github repository.

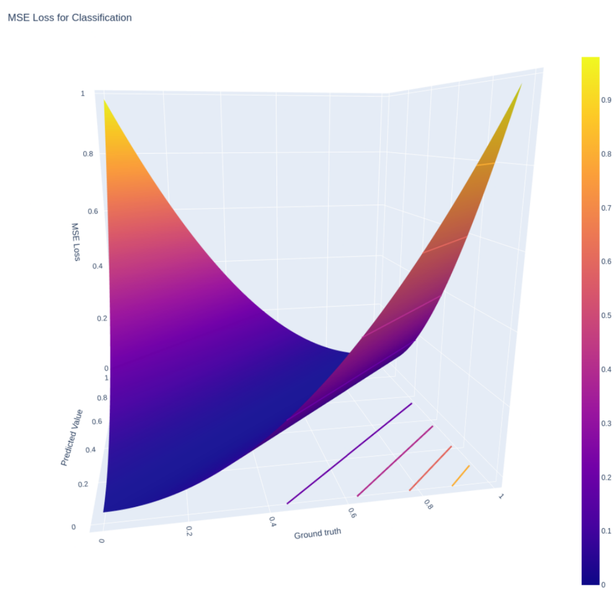
**Question 1:**

a. Cross-Entropy Loss (CE) is utilized in logistic regression tasks to ensure a clear and unambiguous single correct solution.

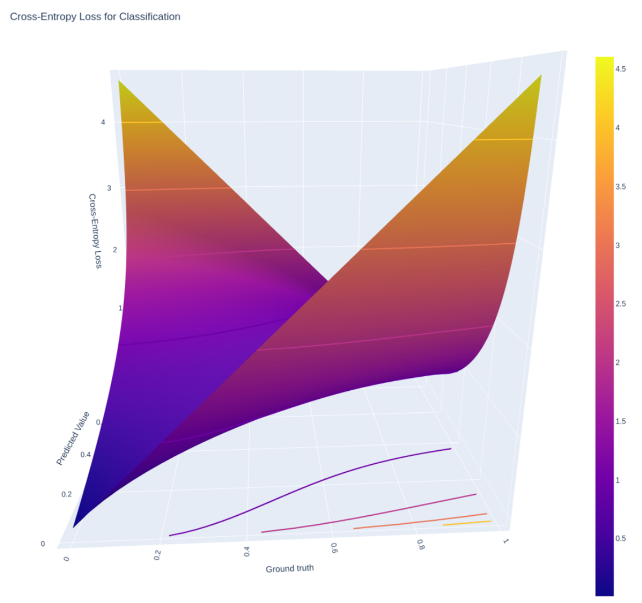
b. Logistic regression is designed to categorize data points into two distinct classes, and the adoption of a single best answer ensures the model's prediction clarity.

c. i.Mean Squared Error (MSE) focuses on minimizing the squared distance between predicted probabilities and actual values (0 or 1). However, in logistic regression, even a small predicted probability significantly distant from the actual class (0 or 1) denotes a substantial classification error.
ii. Cross entropy mitigates these issues by imposing heavier penalties on larger errors in predicted probabilities, particularly when the predicted probability deviates significantly from the actual class (0 or 1).

d. The process of model training along with visualization can be undertsood by the Figure 1.



(a) Mean-Squared Error Loss Surface



(b) Cross-Entropy Loss Surface.

Figure 1: Comparison of MSE and CE Loss Surfaces: The gradient at points (1, 0) and (0, 1) appears lower on the MSE loss surface compared to the CE loss surface. Additionally, the MSE loss surface exhibits lower gradient and slower convergence, while the CE loss surface demonstrates steeper gradient and faster convergence.

Thus, cross-entropy loss function works best for the logistic regression, and its significance can be understood from Figure 1.

**Question 2:**

Neither Cross-Entropy (CE) loss nor Mean Squared Error (MSE) loss guarantees a convex optimization problem.

1. **CE Loss:** The CE loss function for binary classification is given by:

$$\text{CE}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

where $y$ is the true label, $\hat{y}$ is the predicted probability, and $N$ is the number of samples. The CE loss is not convex because of the presence of the logarithmic terms, which lead to non-convexity.

2. **MSE Loss:** The MSE loss function for binary classification is given by:

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

where the terms $(y_i - \hat{y}_i)^2$ introduce non-linearity, making the loss function non-convex.

Since neither CE loss nor MSE loss guarantees convexity, the correct answer is **None**.

**Question 3:**

Consider a model designed for classifying images of digits, such as those found in the **MNIST** [2] dataset available in the **sklearn** [4] library. This dataset comprises a total of 1797 8x8 images of digits. The **MNIST** data set which has been used is taken from the preloaded data set of colab.

a. **Architecture:** The model has 2 hidden layers
 - The hidden layers have Dropout regularization [3] applied to them (during training)
 - The hidden layers can have "relu" or "tanh" activations
 - The number of neurons in each hidden layer is a tuneable hyperparameter

b. The pre-processing strategies employed are:
 - Flatten: As the dense layers expect a 1-dimensional array as input, we flatten the 2-dimensional image into a 1-D array before passing into the dense layer
 - Scaling: Each element of the input array is a number between 0..16 (as described in the dataset documentation). To bring the scale of the inputs to 0..1, we divide each element by 16.

c. The model has been established to optimize the hyper-parameters. Table 1 displays the hyper-parameters under consideration along with the range of values they are being tuned upon. Additionally, Table 2 presents the combination of the best hyper-parameters obtained through tuning.

| **Hyper-parameter** | **Range of Values** |
|---|---|
| Hidden Layer 1 size | [128, 256, 384, 512, 1024] |
| Hidden Layer 2 size | [16, 20, 24, 28, 32, 64, 128] |
| Learning rate in Adam | [0.1, 0.01, 0.001, 0.0001, 0.00001] |
| Dropout rate | [0.0, 0.1, 0.2, 0.3] |
| Activation function | [ReLU, Tanh] |

Table 1: The table presents the hyper-parameters along with the corresponding range of values they have been tuned upon. Same learning dropout and activation function have been used for all the layers.

| **Hyper-parameter** | **Optimum Value** |
|---|---|
| Number of neurons in Layer 1 | 384 |
| Number of neurons in Layer 2 | 24 |
| Learning rate | 0.01 |
| Dropout rate | 0.0 |
| Activation function | ReLU |

Table 2: The table presents the hyper-parameters along with their optimum values. Same learning dropout and activation function have been used for all the layers. KerasTuner [4] has been used for the hyper-parameter tuning.

**Question 4:**

Due to a shortage of computational resources, we have only trained the models for 10 epochs.

**LeNet-5[1] :**

Progression Over Epochs: LeNet-5 demonstrates a consistent enhancement in accuracy, rising from 17% to 84% over the course of 10 epochs. This improvement is noteworthy considering the simplicity of its architecture at the time of its development. However, its relatively basic structure may restrict its capability to capture intricate features present in the SVHN dataset compared to more complex models.

**AlexNet [1] :**

Performance Stability: Throughout the training duration, AlexNet maintains a stable accuracy of 18%. This lack of improvement suggests that despite its deeper architecture compared to LeNet-5, it may struggle to effectively handle the complexity and variability inherent in the SVHN dataset, possibly due to limitations in how it processes image features.

**VGG-16[2] :**

Inconsistent Performance: VGG-16 exhibits erratic behavior with spikes in loss values, indicating potential issues like overfitting, suboptimal learning rates, or data preprocessing discrepancies specific to the SVHN dataset. Despite its reputation for success in large-scale image recognition, its deeper architecture may not seamlessly adapt to the unique characteristics of SVHN without substantial tuning.

**ResNet-18[3] :**

Consistent Improvement: ResNet-18 displays a robust performance, starting at 71% accuracy in the initial epoch and steadily progressing to 91% by the 10th epoch. Its utilization of residual connections likely mitigates the vanishing gradient problem, enabling effective learning from the SVHN dataset and making it well-suited for this task.

**ResNet-50[4] and ResNet-101[5] :**

Diminishing Returns: While both ResNet-50 and ResNet-101 exhibit notable enhancements over time, their performances do not surpass that of ResNet-18. This could be attributed to the increased complexity of these models, potentially leading to challenges in training effectively on the limited subset of the SVHN dataset used. It suggests that beyond a certain depth, additional layers may not yield proportional benefits for this specific task.

## References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[2] Y. LeCun, C. Cortes, C. Burges, et al. Mnist handwritten digit database, 2010.

[3] S. Nitish. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1, 2014.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[1] https://shorturl.at/opPV0
[2] https://shorturl.at/ckrt8
[3] https://shorturl.at/fmwX1
[4] https://shorturl.at/bkJQW
[5] https://shorturl.at/mKL27