1 ASSIGNMENT 1

Q1:Which loss function, out of Cross Entropy and Mean Squared Error, works best with logistic regression because it guarantees a single best answer (no room for confusion)? Explain why this is important and maybe even show how it affects the model's training process.
Answer:

- Cross Entropy loss works best with logistic regression because Cross Entropy loss is specifically for classification tasks like logistic regression, ensuring clear optimization towards a single optimal solution. Mean Squared Error, on the other hand, is not well-suited for classification problems and may lead to suboptimal results in logistic regression tasks.
- In logistic regression, the output is a probability estimate between 0 and 1.Cross Entropy loss typically results in faster convergence and better generalization performance compared to MSE, especially in scenarios where the classes are imbalanced or when the decision boundary is non-linear
- Logistic regression helps us classify things into groups, like deciding if an email is spam or not. We use Cross Entropy to measure how close our guesses are to the truth—its like a scorecard. The Cross Entropy ensures we are always moving in the right direction to improve our guesses, like a map guiding us. Using Mean Squared Error instead would be like measuring the length of a road with a thermometer—it which doesn't make any sense. Cross Entropy helps our guesses be more accurate and ensures smooth training without confusion.

Q2:For a binary classification task with a deep neural network (containing at least one hidden layer) equipped with linear activation functions, which of the following loss functions guarantees a convex optimization problem? Justify your answer with a formal proof or a clear argument. (a) CE (b) MSE (c) Both (A) and (B) (d) None
Answer: Answer is option C:

- For the Cross Entropy (CE) Loss Function:

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

  Each term in the loss Lce corresponds to the negative logarithm of the predicted probability of the correct class Since the activation functions are linear, the predicted probabilities $\hat{y_i}$ are linear functions of the input X .The function log(x) is known to be convex for $x > 0$. This convexity holds for each term in the CE loss function.To formally prove convexity, we compute the second derivative of log(x) with respect to x, which is $\frac{1}{x^2} > 0$ and this is always positive for $x > 0$,i.e each term in the CE loss function is convex, and since it is a sum of convex functions, the overall CE loss function is convex.
- The MSE loss function computes the mean of squared errors between the true labels y_i and the predicted values $\hat{y_i}$. Squaring a linear function results in a convex function.Since the squared error $(y_i - \hat{y_i})^2$ is convex, and the sum of convex functions is convex, the overall MSE loss function is convex.

Q3:Dense Neural Network: Implement a feedforward neural network with dense layers only. Specify the number of hidden layers, neurons per layer, and activation functions. How will you preprocess the input images? Consider hyperparameter tuning strategies.
Answer:

- Model Architecture: The model consists of a feedforward neural network with dense layers only. It has two hidden layers with 128 and 64 neurons respectively, followed by an output layer with 10 neurons for classification (since we have 10 classes in the MNIST dataset). ReLU activation function is used for the hidden layers to introduce non-linearity, and softmax activation is used in the output layer for multi-class classification to obtain class probabilities.
- Preprocessing: The input images are preprocessed by first flattening them to vectors of size 28*28 (= 784) since dense layers require 1D input. Then, pixel values are scaled to the range [0, 1] using Min-Max scaling. This helps in faster convergence during training and better performance.

- Hyperparameter Tuning Strategies: Hyperparameters like the number of hidden layers, neurons per layer, activation functions, and optimizer can be tuned using techniques like grid search, random search, or Bayesian optimization. Additionally, techniques like dropout regularization can be applied to prevent overfitting. It's also essential to monitor the model's performance on a validation set during training to avoid overfitting and select the best-performing model. (code)

Q4:Build a classifier for Street View House Numbers (SVHN) (Dataset) using pretrained model weights from PyTorch. Try multiple models like LeNet-5, AlexNet, VGG, or ResNet(18, 50, 101). Compare performance comment why a particular model is well suited for SVHN dataset.
Answer:

- Data Preparation: We loaded the SVHN dataset and applied standard preprocessing techniques, including resizing, normalization, and converting images to tensors.
- Model Selection: We considered various pretrained models available in PyTorch, including LeNet-5, AlexNet, VGG (11 and 16), and ResNet (18, 50, and 101).
- Training and Evaluation: Each model was fine-tuned on a subset of the training data and evaluated on a validation set. We used Adam optimizer with a learning rate of 0.001 and CrossEntropyLoss as the loss function. The models were trained for 5 epochs.(code)