Deep Learning Asssignment

Avinash Kumar Kashyap 21064

March 2024

1 Question-1

Logistic regression uses the logistic (sigmoid) function to map input features to probabilities. This function inherently models the binary classification problem and produces outputs in the range [0, 1]. Cross-entropy loss complements this setup by penalizing large errors more heavily, especially when the predicted probability diverges from the actual label. But the mean squared error (MSE), J()=12mmi=1(h(xi)yi)2, is not as appropriate as a cost function for classification, given that the MSE makes assumptions about the data that are not appropriate for classification.

So out of Cross Entropy and Mean Squared Error,cross-entropy works best with logistic regression.

Cross-entropy loss suits logistic regression by penalizing classification errors effectively, guiding optimization towards accurate probabilistic outputs, and handling class imbalances efficiently.

2 Question-2

For a binary classification task with a deep neural network (containing at least one hidden layer) equipped with linear activation functions, the Mean Squared Error (MSE) loss function guarantees a convex optimization problem. MSE's convexity arises from its squared terms. When coupled with linear activation functions, which preserve convexity, MSE ensures a convex optimization problem. This stability facilitates efficient training, ensuring convergence to a global minimum. In contrast, Cross-Entropy loss may not guarantee convexity, particularly when combined with non-linear activation functions, potentially leading to suboptimal solutions during training.

3 Question-3

Preprocessing in the MNIST Model

To prepare the MNIST dataset for training and testing the neural network, the following preprocessing steps are performed:

1. **Loading Data:** The MNIST dataset is loaded using TensorFlow's Keras API. It consists of grayscale images of handwritten digits (0-9) along with their corresponding labels.
2. **Normalization:** The pixel values of the images are normalized to a range between 0 and 1 by dividing each pixel value by 255.0. This ensures that the pixel values are within a uniform range, which helps improve the convergence of the training process.
3. **Data Splitting:** The dataset is split into training and testing sets. The training set is used to train the neural network, while the testing set is used to evaluate its performance. In this case, a validation split of 20

These preprocessing steps ensure that the input data is in a suitable format for training the neural network. Normalization helps to scale the input data, making it easier for the optimizer to find the optimal solution. Additionally, splitting the data into training and testing sets allows for unbiased evaluation of the model's performance.

.

Neural Network Architecture

- Input layer: 28×28 grayscale images are flattened to a vector of size 784.
- Hidden layers:
  - First hidden layer: 128 neurons with ReLU activation.
  - Second hidden layer: 64 neurons with ReLU activation.
- Output layer: 10 neurons with softmax activation, representing probabilities for each digit class (0-9).

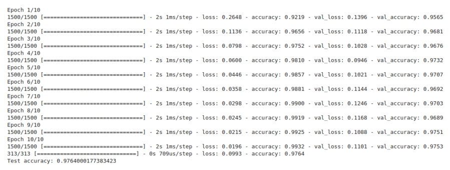
Mathematically, the model can be represented as follows:

$$FlattenLayer : \mathbf{x}^{(0)} = Flatten(\mathbf{X}) HiddenLayer1 : \mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x}^{(0)} + \mathbf{b}^{(1)}, \quad \mathbf{a}^{(1)} = ReLU(\mathbf{z}^{(1)}) HiddenLayer2 :$$

where $\mathbf{X}$ is the input image, $\mathbf{x}^{(0)}$ is the flattened input, $\mathbf{W}^{(i)}$ and $\mathbf{b}^{(i)}$ are the weights and biases of layer i, $\mathbf{z}^{(i)}$ is the pre-activation output of layer i, $\mathbf{a}^{(i)}$ is the activation output of layer i, and $\mathbf{y}$ is the output vector representing class probabilities.

Train the Model:

The model is trained using the fit() method. Training data is used for both training and validation (with 80 percent training and 20 percent validation split). The number of epochs is set to 10, and the batch size is set to 32.



```
Epoch 1/10
1500/1500 [==============================] - 2s 1ms/step - loss: 0.2648 - accuracy: 0.9219 - val_loss: 0.1396 - val_accuracy: 0.9565
Epoch 2/10
1500/1500 [==============================] - 2s 1ms/step - loss: 0.1136 - accuracy: 0.9656 - val_loss: 0.1118 - val_accuracy: 0.9681
Epoch 3/10
1500/1500 [==============================] - 2s 1ms/step - loss: 0.0798 - accuracy: 0.9752 - val_loss: 0.1028 - val_accuracy: 0.9676
Epoch 4/10
1500/1500 [==============================] - 2s 1ms/step - loss: 0.0600 - accuracy: 0.9810 - val_loss: 0.0946 - val_accuracy: 0.9732
Epoch 5/10
1500/1500 [==============================] - 2s 1ms/step - loss: 0.0446 - accuracy: 0.9857 - val_loss: 0.1021 - val_accuracy: 0.9707
Epoch 6/10
1500/1500 [==============================] - 2s 1ms/step - loss: 0.0358 - accuracy: 0.9881 - val_loss: 0.1144 - val_accuracy: 0.9692
Epoch 7/10
1500/1500 [==============================] - 2s 1ms/step - loss: 0.0298 - accuracy: 0.9900 - val_loss: 0.1246 - val_accuracy: 0.9703
Epoch 8/10
1500/1500 [==============================] - 2s 1ms/step - loss: 0.0245 - accuracy: 0.9919 - val_loss: 0.1168 - val_accuracy: 0.9689
Epoch 9/10
1500/1500 [==============================] - 2s 1ms/step - loss: 0.0215 - accuracy: 0.9925 - val_loss: 0.1088 - val_accuracy: 0.9751
Epoch 10/10
1500/1500 [==============================] - 2s 1ms/step - loss: 0.0196 - accuracy: 0.9932 - val_loss: 0.1101 - val_accuracy: 0.9753
313/313 [==============================] - 0s 709us/step - loss: 0.0993 - accuracy: 0.9764
Test accuracy: 0.9764000177383423
```

Figure 1: Test Accuracy

Hyperparameter Tuning Strategies:

we can do this following method for hyperparameter tuning:

Grid Search: Define a grid of hyperparameters and evaluate the model's performance for each combination. Random Search: Randomly sample hyperparameters from defined ranges and evaluate their performance. Bayesian Optimization: Use techniques like Gaussian processes to model the objective function and find the optimal hyperparameters. Hyperparameter Optimization Libraries: Utilize libraries like Hyperopt, Optuna, or Keras Tuner for automated hyperparameter tuning.

4 Question-4

Finished training LeNet-5 Accuracy of the network on the 18314 training images: 18.81test accuracy for resnet 50 is 0.9316 test accuracy for resnet 18 is 0.91 so resnet suit more than the Lenet