Deep Learning Report

March 27, 2024

Khushi Chaudhari
21084

1 QUESTION 1

- In supervised learning, cross-entropy loss is typically employed for classification tasks, whereas mean squared error (MSE) is commonly used for regression problems. Logistic Regression, a technique utilized for binary classification, outputs probabilities within the range of [0,1], indicating the likelihood of an input belonging to a specific class.
  Utilizing cross-entropy loss with logistic regression offers a more nuanced depiction of the disparity between the actual and predicted classes. Conversely, MSE might only yield values such as 0, 1, or -1 as losses, lacking the granularity to precisely gauge the extent of misclassification for individual examples.

2 QUESTION 2

- When the activation function is linear, the neural network essentially functions as a linear classifier. Consequently, employing cross-entropy loss creates a convex optimization problem. Following the reasoning in the previous question, utilizing cross-entropy loss enables us to assess the extent of misclassification for each example and subsequently average these discrepancies, ensuring a clear path toward the optimal solution.

  Conversely, with mean squared error (MSE), the binary classification only indicates whether an example is classified correctly or not. Consequently, there isn't a unified objective guiding optimization for incorrectly classified examples. This lack of a cohesive goal results in the presence of multiple local minima in the optimization landscape.

3 QUESTION 3

- In this question,a feedforward neural network (with dense layers only) has been implemented which resulted in an accuracy of 97.75 percent. The code attached defines a neural network architecture using TensorFlow/Keras to classify handwritten digits from the MNIST dataset with the following specifications:
  • Number of Hidden Layers: The model consists of three hidden layers.
  • Neurons per Layer:
  First hidden layer: 256 neurons Second hidden layer: 128 neurons Third hidden layer: 64 neurons
  • Activation Functions:
  The activation function used for all hidden layers is ReLU (Rectified Linear Unit). The output layer uses the softmax activation function since this is a multi-class classification problem.
  The images were preprocessed in the following way:
  • Reshaping: The input images from the MNIST dataset are reshaped to flatten them into a one-dimensional array. Originally, the images are 28x28 pixels, and by reshaping them, each image becomes a vector of length 784 (28*28).

• Normalization: After reshaping, the pixel values of the images are normalized. This is done by dividing each pixel value by 255.0. Since the pixel values range from 0 to 255 (grayscale images), dividing by 255.0 scales the values to the range [0, 1], making them suitable for training neural networks.
• One-Hot Encoding: The target labels (y-train and y-test) are one-hot encoded using the to-categorical function from Keras.
This converts categorical integer labels into a binary matrix, where each row corresponds to a label, and the corresponding column for that label is set to 1 while others are 0.
These preprocessing steps ensure that the input data is properly formatted and scaled before being fed into the neural network model for training and testing. The hyperparameter tuning that has been done in the code includes:
• Number of Epochs: The model is trained for a fixed number of epochs (10 epochs) using the epochs parameter in the fit method.
• Batch Size: During training, the data is divided into batches, and each batch is used to update the model's weights. The batch size is set to 128 using the batch-size parameter in the fit method.
• Optimizer: The Adam optimizer is used for training the model. It's a popular optimization algorithm known for its adaptive learning rate properties. The optimizer is specified as 'adam' when compiling the model.
• Loss Function: Categorical cross-entropy is used as the loss function, suitable for multi-class classification tasks. This is specified as the loss function when compiling the model.
• Validation Data: The validation data is specified using the validation-data parameter in the fit method.
This allows monitoring the model's performance on a separate validation set during training. These are the primary hyperparameters that have been explicitly set or specified in the code. Other hyperparameters, such as learning rate, regularization strength, layer-specific parameters, etc., are left with their default values.

4 QUESTION 4

- In this question, a classifier was built for Street View House Numbers (SVHN) (Dataset) using pretrained model weights from PyTorch.Multiple models like LeNet-5, AlexNet, VGG, or ResNet(18, 50, 101) have been used. Here is the accuracy scores attained by each model on 25 percent of the SVHN dataset:

MODELS	**ACCURACY**
AlexNet	18.8
LeNet-5	19.6
VGG	88.6
ResNet-18	77.1
ResNet-50	30.9
ResNet-101	27.2

Table 1: Accuracy scores of different models

From the above table, we can clearly see that VGG outperforms all the other models, while ResNet models perform fairly better than AlexNet and LeNet-5.
The VGG and ResNet models perform better due to various reasons as mentioned below:
• VGG, ResNet-50, and ResNet-101 are deeper architectures compared to AlexNet, LeNet-5, and ResNet-18. The increased depth allows these architectures to learn more complex features and hierarchies, which can be beneficial for datasets like SVHN that contain diverse digit images in various orientations, scales, and lighting conditions.
• Also, VGG, ResNet-50, and ResNet-101 are known for their parameter efficiency compared to AlexNet, LeNet-5, and even ResNet-18. They achieve this efficiency through techniques like residual connections (in ResNets) and smaller filter sizes (in

VGG).
• Moreover, VGG has been widely used and pre-trained on large-scale datasets like ImageNet. Fine-tuning a pre-trained VGG model on SVHN can leverage the generalization capabilities learned from ImageNet, potentially leading to better performance on SVHN compared to architectures like AlexNet or LeNet-5.