Deep Learning Report - Gurleen Kaur 21120

March 27, 2024

Answer1

For logistic regression, the Cross-Entropy loss function (also known as Log Loss) is more suitable and works best compared to Mean Squared Error (MSE). The primary reason for this suitability lies in the nature of the logistic regression model and the output it predicts, which is a probability that the given input point belongs to a particular class. The Cross-Entropy loss function is specifically designed to measure the performance of a classification model whose output is a probability value between 0 and 1. For logistic regression, the Cross - Entropy loss function is more suitable as compared to Mean Squared Error as logistic regression works on probability ranging from 0 to 1 for a classification problem and cross - entropy loss function is specifically designed to measure the performance of a classification model whose output s probability value between 0 and 1. Mean squared error is used in regression analysis. The gradient of Cross-Entropy loss function with respect to weights leads to faster and more reliable convergence for logistic regression. This is because Cross-Entropy produces gradients that are not dependent on the magnitude of the error, which avoids the problem of vanishing gradients that can occur with MSE when the model's output is close to 1 or 0. With MSE, small errors can lead to very small gradients, significantly slowing down the learning process during training. For binary classification problem formula of Cross-Entropy loss function is: - The MSE loss function for binary

$$\text{Cross-entropy loss} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$$

Figure 1: Cross Entropy Formula

classification is given by: In logistic regression, the model outputs are interpreted as probabilities. Cross-Entropy effectively

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

Figure 2: Mean Square Error Formula

quantifies the discrepancy between the predicted probabilities and the actual distribution, where the true class is assigned a probability of 1, and all others are set to 0. This alignment makes Cross-Entropy an especially suitable metric for evaluating how well a logistic regression model performs. In conclusion Cross-Entropy Loss function is more suitable than Mean Squared Error.

Answer2

The correct answer is (d) None. Neither Cross-Entropy (CE) nor Mean Squared Error (MSE) guarantees a convex optimization problem for a deep neural network with linear activation functions across its layers for the following reasons:

Cross-Entropy (CE) Loss: CE loss measures the difference between two probability distributions, typically used for classification tasks. For a binary classification with a linear activation, the output of the DNN is a linear function of the input. However, applying CE directly to these outputs does not make sense since CE requires output values that represent probabilities (i.e., in the range [0, 1]). Logistic regression, a linear model with a logistic (sigmoid) activation function on the output layer, uses CE because the sigmoid function ensures outputs in the [0, 1] range, compatible with probability interpretations. Without a nonlinear activation function like sigmoid to ensure the model's output is a valid probability, using CE does not guarantee a convex optimization problem in the context of a DNN with linear activations.
Mean Squared Error (MSE) Loss: MSE measures the average squared difference between the estimated values and the actual value. For a binary classification task modeled as a regression problem (which is conceptually possible but not typical with MSE), the optimization landscape's convexity with MSE as the loss function depends on the model's structure. A single-layer neural network (i.e., linear regression) with MSE loss leads to a convex optimization problem. However, for a DNN with multiple layers and linear activations, while the overall transformation from input to output remains linear (thus, one might initially think the problem remains convex with MSE), the introduction of multiple parameters associated with the depth (i.e., weights and biases at different layers) complicates the issue. The optimization problem's convexity is not guaranteed due to the model's inherent structure introducing non-convexities in parameter space, even if the final input-output mapping is linear.

Answer3

In this question, a feedforward neural network featuring dense layers was made to categorize handwritten digits from the MNIST dataset, achieving an accuracy of 97.75. The neural network's structure, defined using TensorFlow/Keras, is characterized by:
Three Hidden Layers : The network includes three hidden layers, structured as follows:

- The first hidden layer contains 256 neurons.
- The second hidden layer comprises 128 neurons.
- The third hidden layer is made up of 64 neurons.

Activation Functions : All hidden layers utilize the ReLU (Rectified Linear Unit) activation function. For the output layer, the softmax activation function is applied, accommodating the multi-class classification nature of the task.
The preprocessing steps applied to the MNIST dataset images were crucial for optimal neural network performance, involving:

- Reshaping: The 28x28 pixel images as input are reshaped into one-dimensional arrays of 784 elements (28*28), transforming the two-dimensional images into vectors.
- Normalization: The reshaped images undergo normalization by dividing each pixel value by 255.0, scaling the values to a [0, 1] range suitable for neural network training.
- One-Hot Encoding: Target labels for both training and testing sets are transformed into a binary matrix representation using Keras's to-categorical function, facilitating the multi-class classification.

Hyperparameter tuning was meticulously addressed with the following specifications:

- Epochs: The model undergoes training over 10 epochs, determining the number of complete passes through the training dataset.
- Batch Size: Set at 128, this parameter dictates the number of samples processed before the model is updated.
- Optimizer: The Adam optimizer, known for its efficiency and adaptive learning rate capabilities, is chosen for this task.
- Loss Function: The model uses categorical cross-entropy as its loss function, fitting for tasks involving multiple classes.
- Validation Data: Specifying validation data during the training phase allows for the performance evaluation on a separate dataset, offering insights into the model's generalizability.

These elements represent the key hyperparameters and settings established or selected in the coding framework, with other parameters such as learning rate and regularization strength remaining at their default values.

Answer 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:

Model	Accuracy
AlexNet	18.79
LeNet-5	82.29
VGG	91.17
ResNet-18	85.28
ResNet-50	88.42
ResNet-101	87.30

Table 1: Result Table

In the comparison, VGG stands out as the top-performing model, surpassing the others, with the ResNet models also showing superior performance to both AlexNet and LeNet-5. This superior performance of VGG and ResNet models can be attributed to several factors:

- VGG, along with ResNet-50 and ResNet-101, features deeper architectures compared to the relatively shallower AlexNet, LeNet-5, and ResNet-18. This added depth enables them to capture more complex and nuanced features within the images, which is particularly advantageous for challenging datasets like SVHN, characterized by its diverse array of digit images varying in orientation, size, and illumination.
- Additionally, VGG and the more advanced ResNet versions exhibit a greater efficiency in their use of parameters. This efficiency stems from employing residual connections (a hallmark of ResNet models) and utilizing smaller convolutional filters (a characteristic of the VGG architecture), which enhances their ability to learn from the data without a proportional increase in computational demand.
- Furthermore, the widespread availability of VGG models pre-trained on extensive datasets such as ImageNet offer a significant advantage. When these pre-trained models are fine-tuned on specific datasets like SVHN, they can apply the broad and versatile feature-recognition capabilities acquired from ImageNet to excel in new, related tasks, outperforming models like AlexNet and LeNet-5 that lack this extensive pre-training background.