DSE316-Deep Learning Assignment Report-Shalvika Srivastav-21247

27/3/24

1 QUESTION 1

Cross-entropy loss function is typically preferred over mean squared error (MSE) for logistic regression because it's specifically designed for classification tasks, while MSE is more suitable for regression tasks.

In logistic regression, the output of the model is a probability value between 0 and 1, representing the likelihood of belonging to a certain class. Cross-entropy loss, also known as log loss, quantifies the difference between the predicted probabilities and the actual binary labels. It penalizes incorrect classifications more heavily, especially when the predicted probability diverges significantly from the true label.Using cross-entropy loss ensures that the model aims to predict probabilities that are as close as possible to the actual labels. This is crucial in logistic regression because it guarantees a single best answer by optimizing the parameters of the model to minimize the cross-entropy loss.

The cross-entropy loss function is given by:

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

Where:

- N is the number of samples.
- yi is the true label (0 or 1) for the i-th sample.
- pi is the predicted probability that the i-th sample belongs to class 1.

On the other hand, using mean squared error with logistic regression might not yield optimal results because it doesn't consider the probabilistic nature of the predictions. MSE tends to penalize large errors heavily, which may not be appropriate for a classification task where predictions are probabilities.

In terms of the training process, using cross-entropy loss guides the model to update its parameters in a direction that minimizes the difference between predicted probabilities and actual labels. Additionally, the gradient of the cross-entropy loss function is well-defined, which facilitates efficient optimization using techniques like gradient descent.

Hence,cross-entropy loss is preferred for logistic regression because it is used for classification tasks, encourages the model to output probabilities, and guarantees a single best answer by optimizing the parameters to minimize the loss function.

2 QUESTION 2

In the context of a binary classification task with a deep neural network containing linear activation functions, neither the Cross Entropy (CE) loss nor the Mean Squared Error (MSE) loss guarantees a convex optimization problem. Therefore, the correct answer is (d) None.

Cross Entropy (CE) Loss: The CE loss function for binary classification is given by:

$$\text{Cross-entropy loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right) \tag{2}$$

While the CE loss is convex with respect to model parameters, it is not convex with respect to the predictions. yi(hat). This non-convexity arises from the presence of the logarithmic terms in the loss function. Consequently, optimization algorithms may converge to local minima instead of the global minimum.

Mean Squared Error (MSE) Loss: The MSE loss function for binary classification is given by:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3}$$

Similar to the CE loss, the MSE loss is convex with respect to model parameters but not convex with respect to the predictions yi(hat). The squared term in the loss function introduces non-convexity, leading to multiple local minima.

Due to the non-convex nature of both the CE and MSE loss functions concerning the predictions, neither guarantees a convex optimization problem.

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:

Table 1: Accuracy scores of different models

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

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.