Deep Learning Report

March 27, 2024

Gargi Surendra Yeole, 21110

1 Question 1

- 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.

  Reason why Cross-Entropy is preferred:

- Gradient Behavior: The gradient of the Cross-Entropy loss function with respect to the 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.

- Interpretation of Output: Logistic regression models output probabilities, and Cross-Entropy directly measures the difference between two probability distributions: the predicted probability distribution and the actual distribution (where the actual class has probability 1, and all others have 0). This makes Cross-Entropy a more natural fit for measuring the performance of a logistic regression model. Guarantees a Single Solution: The Cross-Entropy loss function, when used with logistic regression, results in a convex loss surface. This means there is only one minimum to which gradient descent can converge, ensuring a single best solution is found. This property is crucial for training stability and predictability.

- The use of Cross-Entropy in logistic regression is important for ensuring that the model learns efficiently and effectively. Since it provides a direct measure of how well the model's probability predictions match the actual labels, it ensures that the training process is focused on improving the model in a way that is directly relevant to the task of classification. Moreover, the mathematical properties of the Cross-Entropy loss function ensure that gradient descent can be applied effectively, finding the optimal model parameters without getting trapped in local minima or suffering from slow convergence.

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: 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.

- Therefore, 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:

  CE: Is not appropriate without a nonlinear activation function at the output layer to ensure outputs are probabilities. MSE: While seeming potentially suitable due to the linear nature of the model's transformation, does not inherently guarantee convexity in the optimization landscape due to the complex parameter interactions introduced by the network's depth.

3 Question 3

In this question , we discuss the implementation of a feedforward neural network with dense layers only for image classification tasks. The objective is to design a neural network architecture, preprocess input images, and explore hyperparameter tuning strategies to achieve optimal performance.

- Number of Hidden Layers: There are 2 hidden layers
- Neurons per Layer: Input=1024, Hidden layer=256, Output=10
- Activation Functions:Activation functions used for the hidden layer is ReLU (Rectified Linear Unit).

Hyperparameters:

- Grid Search: Perform an exhaustive search over a predefined grid of hyperparameter values and evaluate the model's performance for each combination.
- Random Search: Randomly sample hyperparameter values from predefined distributions and evaluate the model's performance for each sample.
- Bayesian Optimization: Utilize Bayesian optimization techniques to efficiently search for the optimal hyperparameters by modeling the objective function and selecting the next set of hyperparameters based on the model's performance.

Preprocessing:

- Number of Epochs: The model is trained for a fixed number of epochs (5 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.

  Result: We got the accuracy of 97.79 percent.

4 Question 4

- The training was done on Google Colab GPUs, and the entire dataset was used. The training was done for 5 epochs.

Model	Accuracy
AlexNet	18.79
LeNet-5	86.01
ResNet-18	92.82
VGG-16	18.79

Table 1: Model and its accuracy

- ResNet Model Outperform Others: We observe that ResNet model achieve the highest accuracy among all models tested. This can be attributed to the deeper architectures of ResNet models, which allow them to capture more complex features from the SVHN dataset.
- Effect of Model Depth: The performance generally improves with the depth of the model. Deeper models like ResNet-18 achieve higher accuracy compared to shallower models like VGG-16 and AlexNet.
- Transfer Learning Benefits: Pretrained models like VGG, AlexNet, and ResNet are pretrained on ImageNet dataset which contains a wide variety of images including digits. Transfer learning allows these models to leverage the learned features and adapt them to the SVHN dataset, resulting in improved performance.