Deep Learning (316/616)
Assignment 1

March 27, 2024

QUESTION 1

The correct answer is (a) CE, which stands for Cross-Entropy loss.

Cross-Entropy Loss (CE)

The Cross-Entropy loss function, often used in classification tasks, measures the difference between two probability distributions. In the context of binary classification, where we're interested in predicting whether an instance belongs to one of two classes, the Cross-Entropy loss can be defined as:

$$\text{CE} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right)$$

Where:

- N is the total number of samples in the dataset.
- y_i is the true label for the ith sample (either 0 or 1).
- $\hat{y}_i$ is the predicted probability that the ith sample belongs to class 1.

Now, let us discuss why Cross-Entropy loss guarantees a convex optimization problem:

Convexity of CE Loss: The Cross-Entropy loss function is convex with respect to its parameters (weights and biases). This convexity property arises primarily from the logarithmic terms in the loss function. Logarithmic functions are convex, ensuring that the overall loss function remains convex.

Composition of Convex Functions: The Cross-Entropy loss function can be decomposed into several convex functions, such as the logarithmic functions. The summation and scaling operations do not affect convexity. The composition of convex functions preserves convexity, and since CE is a composition of convex functions, it remains convex.

Convex Optimization: Convex loss functions lead to convex optimization problems. In the context of deep learning, where optimization methods like gradient descent are commonly employed to minimize the loss function, convexity is desirable. It guarantees that the optimization process converges to the global minimum (assuming appropriate learning rates and other hyperparameters).

In contrast, Mean Squared Error (MSE) does not guarantee convexity. Although it is convex when considered in isolation with respect to each individual parameter, it's not convex with respect to all parameters together. The

nonlinearity introduced by squaring the errors prevents it from being convex in a multivariate setting.

Impact on training: By minimizing cross-entropy loss during training, the model learns to adjust its weights such that the predicted probabilities closely resemble the actual class labels. This translates to a model that can effectively distinguish between the two classes. Therefore, the correct answer is (a) CE.

QUESTION 2

Out of the choices provided, only (d) None guarantees a convex optimization problem for a binary classification task with a deep neural network containing at least one hidden layer equipped with linear activation functions.

Convex Optimization: A convex optimization problem minimizes a convex function over a convex set. A function is convex if a line segment connecting any two points on the curve lies entirely above or on the curve itself.

Linear Activation Functions: Linear activation functions (f(x) = mx + b) introduce no non-linearity into the network's output. The entire network, from input to output, becomes a linear transformation.

Loss Functions and Convexity:

- **Cross-Entropy (CE):** For binary classification with CE loss, the function involves the logarithm of the sigmoid function. The sigmoid function itself is not convex across its entire domain. Therefore, CE loss is not convex.
- **Mean Squared Error (MSE):** MSE is a quadratic function, which is convex. However, when combined with a linear network, MSE becomes a simple linear function of the squared inputs. This linear function is convex, but it loses the ability to distinguish between classes effectively in a binary classification task.

With linear activation functions, the network lacks the complexity to learn non-linear decision boundaries required for binary classification. While MSE becomes a convex function in this case, it is not suitable for the task because it does not effectively model the separation between the two classes. CE loss, although ideal for binary classification with non-linear activation functions, is not convex due to the involvement of the sigmoid function.Therefore, none of the provided loss functions (CE or MSE) guarantees a convex optimization problem in this specific scenario.

QUESTION 3

The code for the question has been uploaded on the github repository.The code provided implements a dense neural network for classifying handwritten digits from the MNIST dataset using TensorFlow and Keras. Let's analyze the architecture and preprocessing steps:

Preprocessing:

- The MNIST dataset is loaded using mnist.load_data() function from Keras.
- Pixel values of the images are normalized to the range [0, 1] by dividing by 255.0.
- Labels are converted to one-hot encoding using to$_c ategorical() function from Keras$.

Model Architecture:

- **Input Layer:** The input layer flattens the input data to a vector shape. No activation function is applied in this layer.

- **Hidden Layers:**
  Hidden Layer 1: 128 neurons with ReLU activation function.
  Hidden Layer 2: 64 neurons with ReLU activation function.
  Hidden Layer 3: 32 neurons with ReLU activation function.

- **Output Layer:** Output layer with num_classes neurons (10 for MNIST digits) using the softmax activation function.

Hyperparameter Tuning Strategies:

- **Learning Rate:** The choice of learning rate is crucial for training neural networks. You can experiment with different learning rates (e.g., using learning rate schedules, or adaptive optimizers like Adam) to find the one that results in faster convergence and better performance.

- **Batch Size:** The batch size affects the speed and stability of training. Larger batch sizes may lead to faster convergence but require more memory. You can experiment with different batch sizes to find the optimal balance.

- **Number of Epochs:** The number of epochs determines how many times the entire dataset is passed through the network during training. Too few epochs may result in underfitting, while too many epochs may lead to overfitting. You can monitor the training/validation loss and accuracy to decide when to stop training to prevent overfitting.

QUESTION 4

Alexnet: Achieves an accuracy of 95%, which is slightly lower than VGG. AlexNet is one of the pioneering deep convolutional neural network architectures that significantly contributed to the advancement of computer vision tasks. Despite its slightly lower accuracy compared to LeNet-5 and VGG in this case, AlexNet remains a powerful and influential model.

VGG: Achieves an accuracy of 96%, which is slightly lower than LeNet-5 but still performs very well. VGG is known for its deep architecture with small convolutional filters and max-pooling layers. It has been successful in various computer vision tasks due to its simplicity and effectiveness.

LeNet-5: Achieves the highest accuracy of 97%, indicating that its architecture is well-suited for the SVHN dataset. LeNet-5 is a classic convolutional neural network architecture and was one of the pioneering models for handwritten digit recognition. Its relatively simple architecture and efficient design make it suitable for this task.

ResNet18, ResNet50, and ResNet101: Achieve accuracies ranging from 88% to 91%. ResNet architectures are known for their deep structure with residual connections, which help mitigate the vanishing gradient problem and enable training of very deep networks. While ResNet models generally perform well on various image classification tasks, they seem to be slightly less effective than LeNet-5, VGG, and AlexNet on the SVHN dataset.

Inference

- LeNet-5 demonstrates the best performance among the models considered, suggesting that its architecture is well-suited for recognizing digits in the SVHN dataset.

- VGG and AlexNet also perform admirably, showcasing the effectiveness of their deep convolutional architectures.

- ResNet models, while still achieving respectable accuracies, appear to be slightly less effective than the other architectures on this specific task. This might indicate that the benefits of their very deep architectures with residual connections are not as pronounced for this dataset compared to simpler architectures like LeNet-5 and VGG.