Deep Learning Report

Vartika Chauhan

27.03.24

1 Question1

In logistic regression, Cross Entropy loss is preferred over Mean Squared Error (MSE) because it's better suited for classification tasks, especially when we're dealing with binary outcomes like "yes" or "no", "spam" or "not spam". Cross Entropy loss helps us measure how close our predicted probabilities are to the actual labels. Think of it this way: if we predict a "yes" with high probability when it's actually a "no", Cross Entropy loss will penalize this mistake more than MSE would. It gives us a clear picture of how well our model is doing in terms of making confident predictions.

Another reason Cross Entropy loss works well is because it helps our model learn faster and more effectively. It does this by providing gradients (hints for how to adjust our model's parameters) that guide our optimization algorithm in the right direction. With Cross Entropy loss, our model learns to be more confident in its predictions, avoiding the problem of getting stuck when it's uncertain. This is crucial, especially in binary classification, where we want our model to make clear decisions: either "yes" or "no", without hesitation. Overall, Cross Entropy loss helps logistic regression produce clear, confident answers, making it a powerful tool for classification tasks.

2 Question2

For a binary classification task with a deep neural network using linear activation functions, the Mean Squared Error (MSE) loss function ensures a convex optimization problem. This means that when we're training our neural network to make predictions, the MSE loss function guarantees that we're always moving towards the best possible solution without getting stuck in local minima. It's like climbing a hill where the path to the top is smooth and clear.

On the other hand, the Cross Entropy (CE) loss function, commonly used for classification tasks, doesn't guarantee a convex optimization problem. This is because CE loss involves calculating probabilities and uses a logarithm operation, which can introduce bumps and dips along the optimization path. As a result, with CE loss, we might not always reach the best solution, as there could be multiple possible answers that look good but aren't the best overall. So, in the context of our binary classification task with linear activation functions, MSE loss ensures a smoother and more reliable optimization journey compared to CE loss.

3 Question 3

This code trains a neural network to understand handwritten numbers from the MNIST dataset. It starts by getting the dataset, which has pictures of numbers and tells us what each number is. Then, it adjusts the brightness of the pictures so they're easier to work with.

After that, it sets up the neural network using the Sequential API. This network is like a series of steps, starting with layers that can understand the pictures, followed by layers that decide which number is in each picture.

Once the setup is done, the code puts everything together and starts training the model. It looks at the pictures and tries to learn from them for 10 rounds, making small adjustments each time. Along the way, it checks how well it's doing by comparing its guesses to the actual answers.

Finally, the model is tested on new pictures it hasn't seen before to see how accurate it is at recognizing handwritten numbers. The goal is to have a model that can reliably tell what number is in a picture, even if it's one it hasn't seen before.

The create dense nn function defines a neural network with two hidden layers. This is specified by the num hidden layers=2 parameter. Each hidden layer has 128 neurons, as specified by the num neurons=128 parameter. ReLU (Rectified Linear Unit) activation function is used for the hidden layers, and softmax activation function is used for the output layer.

4 Queastion4

This code trains a LeNet-5 neural network to recognize digits from pictures of house numbers in the SVHN dataset. It first gets the dataset and adjusts the pictures so the model can understand them better. To speed up learning, it uses only a portion of the training data. Then, the model learns from the data, making guesses and adjusting itself to get better with each round of training. After training, the model's ability to recognize digits is tested on new pictures it hasn't seen before, and the accuracy of its guesses is calculated. This helps us see how well the model can identify digits in real-world situations.

Performance Comparison: LeNet-5 is compared to other models like AlexNet, VGG, and ResNet. Each model's performance metrics, such as accuracy and loss, are evaluated and compared.

Suitability for SVHN Dataset: LeNet-5 is well-suited for the SVHN dataset due to its relatively simple architecture and the nature of the dataset. SVHN contains digit images similar to the MNIST dataset, for which LeNet-5 was originally designed. The dataset consists of cropped images of digits, making it suitable for LeNet-5's convolutional layers to extract features effectively. Additionally, LeNet-5 has fewer parameters compared to deeper architectures like VGG or ResNet, making it computationally efficient, especially when dealing with limited computational resources or a subset of the dataset.