Deep Learning Assignment

Aryan Jain 21055

March 27, 2024

1 Question 1

Which loss function, out of Cross Entropy and Mean Squared Error, works best with logistic regression because it guarantees a single best answer (no room for confusion)? Explain why this is important and maybe even show how it affects the model's training process.

1.1 Answer

Logistic regression is a classification algorithm that predicts the probability of an event happening (binary outcome: 0 or 1). It uses a sigmoid function to take input linear combination of features (weights and inputs) and predicts probability between 0 and 1. A loss function measures how good or bad the model's predictions are compared to the actual labels.

The best loss function for logistic regression is cross-entropy (log loss) because it penalizes the model more for predicting probabilities far away from the actual label. Mean squared error (MSE), which is used for regression tasks, doesn't work well with logistic regression because it penalizes large deviations from the actual label equally, even for probabilities very close to the correct class.

Cross-entropy is specifically designed for classification problems with binary outcomes. It penalizes the model more for predicting probabilities far away from the actual label (0 or 1), helping the model focus on learning the decision boundary between the classes more effectively.

Using cross-entropy ensures the model is optimized for the task at hand, leading to a better overall solution. It results in a smoother loss landscape, which helps the optimization process by avoiding many local minima and making it easier to find the best solution overall.

2 Question 2

For a binary classification task with a deep neural network (containing at least one hidden layer) equipped with linear activation functions, which of the following loss functions guarantees a convex optimization problem? Justify your answer with a formal proof or a clear argument. (a) CE (b) MSE (c) Both (A) and (B) (d) None

2.1 Answer

In a deep neural network employing linear activation functions, the model's output becomes a linear combination of its inputs and weights, effectively rendering it a linear model.

- **The Mean Squared Error (MSE) loss function** computes the squared difference between predicted values and actual labels. When the model is simplified to a linear function with linear activation functions, the MSE loss function guarantees convex optimization. This convexity arises from the quadratic nature of the loss function.
- **Cross Entropy (CE) Loss Function:** Although the cross-entropy (CE) loss is typically associated with logistic activation functions, it remains convex when used with linear models. This convexity stems from its interpretation as a maximum

likelihood estimation, which is inherently convex. Even when linear activation functions are present, the CE loss retains convexity, ensuring a convex optimization problem.

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

Where: N denotes the total number of samples, y_i represents the actual label corresponding to sample i, p_i stands for the predicted probability (output) associated with sample i.

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

where N is the number of samples, y_i represents the actual label for sample i, and $\hat{y}_i$ denotes the predicted value for sample i.

Hence, both MSE and CE loss functions guarantee convex optimization problems when employed with linear activation functions. Therefore, the correct answer is (c) Both (A) and (B).

3 Question 3

Dense Neural Network: Implement a feedforward neural network with dense layers only. Specify the number of hidden layers, neurons per layer, and activation functions. How will you preprocess the input images? Consider hyperparameter tuning strategies.

3.1 Report

1. Number of Hidden layers : 2
2. Neurons in Input Layer : 64
3. Neurons in 1st Hidden Layer : 128
4. Neurons in 2nd Hidden Layer : 256
5. Neurons in Output Layer : 10

Preprocessing

Preprocessing the input photos can be done using a number of widely used methods, including:

- Normalisation : It is the process of scaling an image's pixel values to a range of 0 to 1.
- Resizing (Flattening): Flattening the input image matrix and resizing the images to a standard size.

Hyperparameter Tuning

Hyperparameter tuning is crucial for optimizing the performance of neural networks. Some common strategies include:

- Grid Search: A comprehensive search using a pre-established set of hyperparameters.
- Random Search: Randomly sampling hyperparameters from predefined distributions.

In conclusion, proper preprocessing of input images and effective hyperparameter tuning strategies are essential for maximizing the performance of dense neural networks.

4 Question 4

Build a classifier for Street View House Numbers (SVHN) (Dataset) using pretrained model weights from PyTorch. Try multiple models like LeNet-5, AlexNet, VGG, or ResNet(18, 50, 101). Compare performance comment why a particular model is well suited for SVHN dataset. (You can use a subset of dataset (25%) in case you do not have enough compute.)

4.1 Introduction

4.2 Methodology

- **Data Loading:** We use a subset (25 percent) of the SVHN dataset, which contains images of house numbers from Google Street View.
- **Pretrained Models:** We select pretrained models from PyTorch's model zoo, including AlexNet, and ResNet-18.
- **Fine-tuning:**To improve performance, the weights of each pretrained model are fine-tuned using the Adam optimizer and cross entropy loss on the SVHN dataset.
- **Evaluation:** We measure the performance of each model using accuracy as the metric.

4.3 Results

1. **AlexNet**: Accuracy = 85.86 %
2. **VGG-11**: Accuracy = 87.44 %
3. **ResNet-18**: Accuracy = 86.85 %

4.3.1 Discussion

After evaluating the models on the SVHN dataset, we observed the following:

- AlexNet: While deeper than LeNet-5, it may still struggle with complex features.
- VGG: With its deeper architecture and smaller filter sizes, VGG-11 performs well in capturing intricate details.
- ResNet: The residual connections in ResNet models make them suitable for very deep networks like SVHN.

In conclusion, VGG-11 is the best-performing model on the SVHN dataset due to its ability to capture complex features effectively.