Assignemnt-1, DSE 316

Ashim Dhor, 21057
Data Science and Engineering

27/03/2024

1 QUESTION-1

In logistic regression, the choice of loss function plays a critical role in the model's effectiveness and reliability. While both Cross Entropy and Mean Squared Error (MSE) are common choices, Cross Entropy is particularly suitable for logistic regression for several compelling reasons, detailed below.

1.1 Suitability for Classification Problems

- **Logistic Regression:** A method used for binary classification that models the probability of inputs belonging to a particular class.
- **Cross Entropy:** Designed for classification tasks, it measures the dissimilarity between actual and predicted probability distributions, making it inherently suited for logistic regression.

1.2 Convexity

- **Cross Entropy and Convexity:** Creates a convex loss surface in logistic regression, ensuring a single global minimum.
- **Importance of Convexity:** Convexity ensures that optimization algorithms, like gradient descent, converge to the global minimum, avoiding local minima that non-convex surfaces, like those potentially created by MSE, might present.

1.3 Alignment with Logistic Regression Objectives

- **Probabilistic Nature:** Logistic regression outputs probabilities, and Cross Entropy aligns with this by effectively penalizing incorrect probabilistic predictions.
- **Penalization of Incorrect Predictions:** Cross Entropy heavily penalizes predictions far from actual values, guiding the model towards accurate probability estimates.

1.4 Gradient Behavior and Model Training

- **Gradient Magnitude:** The magnitude of the gradient under Cross Entropy decreases as the prediction approaches the actual label, aiding in a stable and effective training process.
- **Comparison with MSE:** MSE lacks the beneficial gradient properties for logistic regression, potentially leading to vanishing or exploding gradients.

1.5 Example of Impact on Training Process

Consider a logistic regression model with a true label $y = 1$ and a predicted probability p. The Cross Entropy loss is defined as $-\log(p)$, encouraging the model to increase p towards 1. This aligns with the model's goal of predicting accurate probabilities. In contrast, the MSE, defined as $(1 - p)^2$, also encourages increasing p but does not align as closely with the model's probabilistic outputs and does not penalize wrong predictions as effectively.

Conclusion and Inference

Cross Entropy is the preferred loss function for logistic regression over MSE due to its suitability for classification tasks, the convexity it ensures, its alignment with logistic regression's probabilistic objectives, and the advantageous gradient behavior. These benefits collectively ensure optimal training of the model to predict probabilities accurately, thereby guaranteeing a single, unambiguous solution.

2 QUESTION-2

Given a deep neural network designed for a binary classification task, with at least one hidden layer and equipped with linear activation functions, we examine which loss function, Cross Entropy (CE) or Mean Squared Error (MSE), guarantees a convex optimization problem. The options are: (a) CE, (b) MSE, (c) Both (A) and (B), (d) None.

2.1

Answer: (d) None

2.2

Justification:

Deep neural networks, even with linear activation functions, do not ensure convexity in optimization due to their structure and the interaction between layers. The choice of loss function, whether CE or MSE, does not alter this fundamental aspect.

1. **Deep Neural Networks:** The architecture of deep neural networks introduces complexities that affect the optimization landscape. Even with linear activation functions, the network's behavior is not simplified to guarantee convexity.
2. **Linear Activation Functions:** These functions, when used throughout a deep neural network, lead to a model that behaves as a single linear transformation. However, this does not imply convexity of the optimization problem with respect to the network parameters.
3. **Cross Entropy and MSE:** While CE and MSE have properties that are desirable for certain types of problems (CE for classification and MSE for regression), neither guarantees a convex optimization landscape in the context of deep neural networks with linear activations.

Conclusion:

No loss function, whether Cross Entropy (CE) or Mean Squared Error (MSE), guarantees a convex optimization problem in deep neural networks with linear activation functions. The structural complexities of such networks preclude the convexity of the optimization landscape, leading to the conclusion that the correct answer is (d) None.

3 QUESTION-3

3.1 Introduction

This report outlines the implementation of a feedforward dense neural network designed for image classification tasks. The implementation is realized using TensorFlow with its Keras API, showcasing a model architecture, preprocessing steps, and hyperparameter tuning strategies.

3.2 Model Architecture

The implemented model comprises:

- Input layer that flattens the image matrices to vectors.
- One hidden layer with 512 neurons, utilizing ReLU (Rectified Linear Unit) as the activation function to introduce non-linearity.
- Output layer with 10 neurons (for 10 classes), using softmax activation for probability distribution across class labels.

3.3 Preprocessing Steps

Input images are preprocessed through the following steps:

1. Reshaping: Images are reshaped to ensure a consistent input size.
2. Normalization: Pixel values are normalized to the range [0, 1] by dividing by 255, enhancing model training efficiency.

3.4 Hyperparameter Tuning Strategies

Hyperparameter tuning is essential for optimizing model performance. The strategies include:

- Employing an adaptive learning rate (e.g., Adam optimizer) for efficient gradient descent.
- Experimenting with different numbers of neurons in the hidden layer to find a balance between model complexity and overfitting.
- Adjusting the batch size and epochs based on training performance and computational constraints.

3.5 Code Snippet in LaTeX

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load and preprocess data
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1)).astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float32') / 255
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

# Define the model
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28, 1)))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(train_images, train_labels, epochs=5, batch_size=128)

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f'Test accuracy: {test_acc}')
```

3.6 Conclusion

The implemented dense feedforward neural network, with its specific architecture and preprocessing steps, is designed for effective image classification. Through careful hyperparameter tuning, the model's performance can be significantly optimized for higher accuracy in tasks such as MNIST digit classification.

4 QUESTION-4

4.1 Introduction

This report outlines the process and findings from building a classifier for the Street View House Numbers (SVHN) dataset using various pretrained models in PyTorch, including LeNet-5, AlexNet, VGG, and ResNet variants.

4.2 Methodology

We employed a subset (25%) of the SVHN dataset for this task, considering computational limitations. The dataset was preprocessed and normalized. Models were selected based on their historical significance and varied architecture designs, providing a broad view of performance across different network structures.

4.3 Experiment Setup

The experiment involved fine-tuning the following pretrained models: LeNet-5, AlexNet, VGG, and ResNet variants (18, 50, 101), with the final fully connected layer adjusted to output ten classes corresponding to the digits 0-9 in the SVHN dataset.

4.4 Results

The performance of each model was evaluated based on accuracy. Preliminary results suggest that ResNet models, particularly ResNet50, performed exceptionally well due to their deep residual learning framework, which is effective in handling the complexity and variability of real-world images in the SVHN dataset.

4.5 Discussion

The superior performance of ResNet models can be attributed to their ability to mitigate the vanishing gradient problem, allowing for deeper networks that can learn more complex features without a significant increase in training difficulty.

4.6 Conclusion

This study demonstrates the effectiveness of utilizing pretrained models for image classification tasks on datasets like SVHN, with ResNet50 emerging as the most