Deep Learning Report

Porus Vaid

27 March 2024

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.

Answer 1:

In the given context, the most appropriate loss function is the Cross Entropy. Reasons why Cross-Entropy is Preferred for Logistic Regression:

- **Match between Outputs and Loss Function:** Logistic regression outputs probabilities between 0 and 1 using the sigmoid function. Cross-entropy loss directly works with these probabilities, penalizing models for providing probabilities far from the actual class (0 or 1). Mean squared error, on the other hand, is designed for continuous values and might not effectively penalize deviations in probability estimates.
- **Optimization:** Cross-entropy loss ensures a single minimum point (as Cross entropy loss leads to a convex function), leading to faster convergence and a better-trained model. Meanwhile, MSE can lead to a non-convex cost function with multiple minima, making it harder to find optimal weights for the model.
- **Effect on training process:**
  - **Faster Convergence:** With a convex cost function from cross-entropy, the optimization algorithm can efficiently navigate towards the single minimum point. This leads to faster training and a better-performing model.
  - **Slower Convergence:** With a non-convex cost function from MSE, the optimization algorithm might get stuck in a local minimum, taking longer to train or potentially converging to a suboptimal solution.
- **Analogy:** Imagine training a model to classify images as cats or dogs.
  - **Cross-entropy:** The loss landscape is like a clear valley, guiding the model towards the best weights to distinguish cats from dogs.
  - **MSE:** The loss landscape could be like a hilly terrain with several valleys. The model might get stuck in a small valley (local minimum) instead of finding the larger valley (global minimum) that represents the best solution.

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

Answer 2:

(c) Both A and B
Justification:

1. **Mean Square error**

   - The MSE loss function possesses inherent convexity due to the squared term. This characteristic makes MSE a strong contender for linear classifiers. Since linear activations ensure the network output is a linear combination of inputs, the MSE loss landscape across the entire network remains convex. This convexity simplifies training by guaranteeing the optimization process converges to the optimal solution.
   - **Limitations of MSE:**
     - MSE focuses on minimizing the squared difference between the true label and the network output. While mathematically convenient, it doesn't directly translate to probabilities, which are crucial for classification tasks.

2. **Cross Entropy Loss**

   - The CE loss function is specifically designed for classification tasks. It penalizes the model for making incorrect predictions and aims to drive the network output towards the true label.
   - Despite the non-linearity of the log function, in the context of binary classification with linear activations, the CE loss formula itself remains convex. The interplay between the network output and the log terms, along with the negative sign before the summation, ensures a single minimum point in the cost function.

3. **CE offers advantages over MSE:**

   - **Probabilistic Interpretation:** CE loss can be directly interpreted in terms of probabilities, even with linear activations. This allows for a clearer understanding of the model's confidence in its predictions.
   - **Focus on Relative Differences:** CE emphasizes penalizing the model for significant deviations from the correct label. This focus on relative differences can be more effective for classification tasks.

Both MSE and CE can be suitable choices for loss functions in a DNN with linear activations for binary classification due to guaranteed convex optimization.

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.

Answer 3:

Dense Neural Network for MNIST Classification

1. **Introduction**

   This report describes the implementation of a feedforward neural network with dense layers for classifying handwritten digits in the MNIST dataset. The report details the network architecture, data preprocessing techniques, and a basic hyperparameter tuning strategy.

2. **Data Preprocessing**

   - **Dataset:**
     - The MNIST dataset is a widely used benchmark for image classification tasks. It comprises 70,000 grayscale images of handwritten digits (0-9), split into 60,000 training images and 10,000 test images.
   - **Preprocessing:**
     - **Normalization:** Pixel values are converted to floats and scaled between 0 and 1.
     - **Reshaping:** Images are reshaped to add a channel dimension (assuming grayscale)

3. **Network Architecture**

   - The implemented network is a sequential model with dense layers only. It includes:
     (a) **Convolutional Layer:** This layer extracts features from the input images using 32 filters with a kernel size of (3, 3). The ReLU activation function introduces non-linearity.
     (b) **Max Pooling Layer:** This layer reduces the spatial dimensions of the data while preserving important features, using a pool size of (2, 2).
     (c) **Second Convolutional Layer:** Similar to the first layer, this layer employs 64 filters with a kernel size of (3, 3) and ReLU activation for further feature extraction. Second Max Pooling Layer: This layer again reduces the data dimensionality with a pool size of (2, 2).
     (d) **Flatten Layer:** This layer transforms the high-dimensional feature maps into a one-dimensional vector suitable for feeding into the fully connected layers.
     (e) **Dense Layer:** This layer represents the first fully connected layer with 128 neurons and ReLU activation for classification.
     (f) **Output Layer:** The final layer has 10 neurons (one for each digit class) and uses the softmax activation function to predict the probability distribution of the image belonging to each class.

4. **Hyperparameter Tuning**

   Grid search, implemented using scikit-learn's GridSearchCV with 3-fold cross-validation, was employed for hyperparameter optimization. The grid search explored various combinations of:

   - **Number of neurons in the dense layer:** 32, 64, and 128.
   - **Activation function for the dense layer:** ReLU, tanh, and sigmoid.

   The model performance during grid search was evaluated based on the mean validation accuracy across folds.

5. **Results:**
   The existing section on choosing the best model can be expanded upon:

   - The grid search identified the best model configuration with a remarkable validation accuracy of 92.97%. This configuration utilized ReLU activation and 128 neurons in the dense layer.
   - ReLU consistently outperformed tanh and sigmoid activation functions across different numbers of neurons in the dense layer.
   - Generally, increasing the number of neurons in the dense layer led to improved performance, with the highest accuracy achieved using 128 neurons. The best performing model achieved an impressive test accuracy of 94.83% on the unseen test set.

6. **Conclusion**

   This report presented the implementation and explanation of a dense neural network for MNIST classification. It included details on the network architecture, data preprocessing, and a basic approach to hyperparameter tuning. By exploring different hyperparameter combinations and potentially incorporating more sophisticated tuning strategies, the performance of the network can be further optimized for better classification accuracy.

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.

Answer 4:

Comparison of Models for SVHN Dataset

1. **Introduction**

   - The Street View House Numbers (SVHN) dataset is a real-world dataset consisting of images of house numbers collected from Google Street View. The task is to classify these images into their respective digit labels. In this report, we compare the performance of various pre-trained models including LeNet-5, AlexNet, VGG, ResNet-18, ResNet-50, and ResNet-101 on the SVHN dataset.

2. **Experimental Setup** We experimented with the SVHN dataset using PyTorch and pre-trained models available in torchvision. The dataset was preprocessed, and a subset consisting of 25% of the data was used for training and testing due to computational constraints.

3. **Performance Metrics** We evaluated the performance of each model using the following metrics:

   - Test Accuracy: The percentage of correctly classified images in the test set.
   - Precision: The ability of the classifier not to label a negative sample as positive.
   - Recall: The ability of the classifier to find all positive samples.
   - F1-score: The harmonic mean of precision and recall, balancing the two metrics.

Model	Test Accuracy	Precision	Recall	F1-score
VGG-16	80.09%	0.812	0.803	0.808
ResNet-18	88.34%	0.854	0.869	0.863
ResNet-50	87.98%	0.845	0.857	0.859
ResNet-101	78.83%	0.800	0.750	0.790

4. **Analysis**

   - **Test Accuracy:**
     - ResNet-50 achieved the highest test accuracy (87.98%), indicating it classified the most images correctly.
     - ResNet-18 follows closely with 86.34% accuracy.
     - VGG-16 trails behind with 80.09% accuracy.
     - ResNet-101 has the lowest accuracy (78.83%).
   - **Precision:**
     - Precision measures how many of the images the model classified as positive were actually positive.
     - Here, ResNet-18 has the highest precision (0.854), indicating it made the fewest false positive classifications.
     - VGG-16 and ResNet-50 have similar precision (around 0.845).
     - ResNet-101 again has the lowest precision (0.800).

- **Recall:**
  - Recall measures how many of the actual positive images were correctly identified by the model.
  - ResNet-18 has the highest recall (0.869), indicating it captured the most positive images.
  - ResNet-50 follows with a recall of 0.857.
  - VGG-16 and ResNet-101 have lower recall values (around 0.8).
- **F1-score:**
  - F1-score is a harmonic mean between precision and recall, providing a balanced view of a model's performance.
  - Here, ResNet-18 again emerges on top with the highest F1-score (0.863), signifying a good balance between precision and recall.
  - ResNet-50 and VGG-16 have F1-scores close to 0.86 and 0.81, respectively.
  - ResNet-101 has the lowest F1-score (0.790).