Deep Learning Assignment

Siddhi Pravin Lipare 21262

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

When selecting the appropriate loss function for logistic regression, Cross Entropy (also known as Log Loss) emerges as the superior option. Here's why:

1. **Probabilistic Outputs**: Logistic regression predicts probabilities using a sigmoid function, ensuring outputs between 0 and 1. Unlike Mean Squared Error (MSE), which suits continuous outputs, Cross Entropy penalizes deviations effectively for probability predictions. It aligns better with the probabilistic nature of logistic regression.

2. **Certainty Capture**: Cross Entropy employs the logarithm function, which emphasizes mismatches between predicted probabilities and actual labels. It encourages the model to make clear predictions, pushing probabilities towards 0 or 1. This clarity is crucial in classification tasks, fostering confident predictions.

3. **Single Best Answer Importance**: In classification problems, ambiguous probabilities (e.g., close to 0.5) lead to confusion. Cross Entropy minimization promotes decisive class distinctions, crucial for tasks like spam detection. It ensures the model learns to make confident classifications.

4. **Impact on Training**: During training, Cross Entropy's steeper penalty for ambiguous predictions directs the model towards clear classifications. Unlike MSE, which might tolerate mediocre predictions, Cross Entropy fosters robust models that confidently separate classes.

In summary, Cross Entropy's design harmonizes with logistic regression's probabilistic outputs, promoting clear and accurate classifications. It plays a vital role in training models to make confident predictions, crucial for various classification tasks.

For logistic regression, the **Cross Entropy Loss** (or Log Loss) is defined as:

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

Where:

- N is the number of samples,
- y_i is the actual label for sample i,
- p_i is the predicted probability (output) for sample i.

This loss function ensures that the model aims to make the most confident predictions, minimizing the divergence between predicted probabilities and actual labels.

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 with linear activation functions, the output of the network is a linear combination of its inputs and weights. Therefore, the model essentially becomes a linear model.

- **Mean Squared Error (MSE) Loss Function:** MSE computes the squared difference between the predicted values and the actual labels. Since the model with linear activation functions reduces to a linear function, the MSE loss function guarantees a convex optimization problem. This is because the MSE loss function applied to linear models results in a convex optimization problem due to the quadratic nature of the loss function.

  For the Mean Squared Error (MSE) loss function, it is defined as:

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

  Where:

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

  This loss function also ensures convex optimization, particularly with linear models.

- **Cross Entropy (CE) Loss Function:** Although CE loss is commonly used with logistic activation functions, it still ensures convex optimization when applied to linear models. Cross entropy loss can be interpreted as a form of maximum likelihood estimation, which is convex. Even with linear activation functions, CE loss maintains convexity, ensuring a convex optimization problem.

Therefore, both MSE and CE loss functions guarantee convex optimization problems when used with linear activation functions. Hence, 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

In the code, I have implemented a feedforward neural network with dense layers only.

1. Number of hidden layers: 1
2. Number of neurons in input layer: 256
3. Number of neurons in hidden layer: 128
4. Number of neurons in output layer: 10

3.1.1 Preprocessing Input Images

For preprocessing the input images, several common techniques can be employed, such as:

- Normalisation: Scaling the pixel values of the images to a range between 0 and 1.
- Resizing: Resizing the images to a consistent size, since they are of varying dimensions.
- Data Augmentation: Applying transformations like rotation, flipping, and shifting to increase the diversity of the training data.

These preprocessing steps help in improving the convergence of the model during training and in generalizing better to unseen data. I have implemented normalisation and resizing in my code.

3.1.2 Hyperparameter Tuning Strategies

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

- Grid Search: Exhaustively searching through a predefined set of hyperparameters.
- Random Search: Randomly sampling hyperparameters from predefined distributions.
- Bayesian Optimization: Using probabilistic models to select the most promising hyperparameters based on past evaluations.

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

I've built a classifier for the Street View House Numbers (SVHN) dataset using pretrained model weights from PyTorch. We experiment with multiple models, including AlexNet, VGG, and ResNet (18 and 101), and have compared their performance on the SVHN dataset.

4.2 Methodology

- **Data Loading:** I've loaded the SVHN dataset (Used only 25 percent), which contains images of house numbers collected from Google Street View.
- **Pretrained Models:** I've chosen pretrained models from PyTorch's model zoo, including AlexNet, VGG-11, and ResNet (18, 101).
- **Fine-tuning:** I've refined each pre-trained model on the SVHN dataset utilizing the Adam optimizer with a learning rate set to 0.0003, coupled with Cross Entropy Loss.
- **Evaluation:** I've evaluated the performance of each model using metrics such as accuracy and loss after training the models for 3 Epochs.

4.3 Results

Model	Accuracy	Loss
AlexNet	86.3%	0.53
VGG-11	86.93%	0.38
ResNet-18	87.43%	0.29
ResNet-101	84.98 %	0.60

Table 1: Performance Metrics of Different Models on SVHN Dataset

4.3.1 Discussion

After evaluating the performance of each model on the SVHN dataset, we can make some observations and draw conclusions regarding their suitability for this task:

- AlexNet: While AlexNet has a deep architecture, it may still lack the capacity to learn intricate patterns present in SVHN images.
- VGG: VGG has a deeper architecture with smaller filter sizes, making it more suitable for capturing intricate details in images. However, it may suffer from overfitting due to its large number of parameters.
- ResNet: ResNet's residual connections allow for easier training of very deep networks, making it well-suited for SVHN classification. ResNet-18 gave us the best results in comparison to the rest of the models.

In summary, ResNet-18 emerges as the top performer on the SVHN dataset owing to its deeper architectures and proficiency in effectively capturing complex features.