Deep Learning Assignment (316/616)

Atul Kumar (2321002)

Q1. Loss Function for Logistic Regression

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.

Solution: Logistic regression favors using Cross Entropy Loss instead of Mean Squared Error (MSE) because it is better suited to the stochastic character of logistic regression outputs and has more favorable gradient features. Logistic regression generates probabilities making Cross Entropy Loss suitable for measuring the discrepancy between two probability distributions. However, Mean Squared Error (MSE) is not well-suited for aligning with probabilistic outputs and can produce deceptive findings when applied to probabilities.

The use of Cross Entropy Loss in logistic regression ensures a consistent gradient efficiency preventing the issue of disappearing gradients that might arise when using Mean Squared Error (MSE). This guarantees more efficient updates of parameters during training resulting in quicker convergence. The use of Cross Entropy as a metric for confidence in predictions is advantageous since it places a greater penalty on inaccurate predictions made with high confidence compared to Mean Squared Error (MSE). This effectively steers the model towards making more decisive and accurate classifications resulting in a single unambiguous response.

Q2. Convex Optimization in Deep Neural Networks

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

Solution: (b) None. Regardless of the loss function, a deep neural network with linear activation functions at all levels converges onto a single linear model. Because linear functions are linear functions, their composition is linear. This limits the model's ability to understand complex patterns or non-linearities, which are common in classification tasks.

Convexity and loss functions: When applied with a linear binary classification model, Cross Entropy Loss does not guarantee convex optimization. Cross-entropy (CE) measures probability distribution dissimilarity. If the model's output represents probabilities, a sigmoid or softmax function in the last layer is used for binary or multi-class classification. Linear activations and CE loss may not give convex optimization because the output can transcend the [0, 1] range, making CE loss indeterminate.

MSE Loss: Mean Squared Error Loss creates a convex optimization problem in linear regression but not in classification problems, even with linear activations. In binary classification, targets are labeled 0 or 1. Using mean squared error (MSE) loss with a deep neural network and linear activations does not ensure convex optimization. Even with linear activations, classification problems and deep network architectures might have many local minima due to non-convexity. Hidden layers, even with linear activations, add complexity to the model which may cause non-convex optimization. Multiple layers and weights in CE and MSE losses provide local minima, saddle points, and potentially flat regions, causing a non-convex optimization problem.

Q3. Dense Neural Network Implementation

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.

Solution: The model consists of two hidden layers: the first with 512 neurons and the second with 256 neurons. Both use the ReLU activation function. The output layer employs the softmax activation function, suitable for multi-class classification tasks.

Preprocessing includes flattening the 28x28 pixel images into 784-dimensional vectors and normalizing pixel values by dividing by 255.

Hyperparameter tuning strategies include varying the learning rate, adjusting batch size, using a sufficient number of epochs with early stopping, experimenting with layer depth and width, employing regularization techniques like L2 regularization and dropout, and exploring different optimizers such as SGD with momentum or RMSprop.

Q4. Classifier for SVHN using Pretrained Models

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 and comment on why a particular model is well suited for the SVHN dataset.

Solution: To construct classifiers for the SVHN dataset using several pretrained models in PyTorch, we adopted a systematic method. This included data preparation, model adaptation, training, evaluation, and performance comparison.

Data Preparation

The SVHN dataset consists of color photographs of house numbers collected in a natural environment. Each image is annotated with the digit it represents. Images were resized to fit the input requirements of each pretrained model (e.g., 224x224 for AlexNet and VGG), and normalized to match the mean and standard deviation values used for ImageNet.

Model Adaptation

Models were adapted for the SVHN dataset by modifying the last layer to have 10 outputs, corresponding to the 10 digits, and initializing the weights of this new layer.

Training and Evaluation

Each model was trained on the SVHN training set using Cross Entropy Loss and an SGD or Adam optimizer. Techniques such as learning rate scheduling and early stopping were employed to enhance training efficiency and prevent overfitting.

Performance Comparison

Models were assessed based on accuracy, training duration, and complexity. Deeper models like ResNet were found to be more effective due to their ability to capture more complex information through residual connections, which is beneficial for the diverse and intricate backgrounds in SVHN images. Simpler models like LeNet-5, while faster to train, may not capture as detailed features as deeper models but could be sufficient for less complex tasks.

Conclusion

ResNet models, with their ability to capture deeper features, showed superior performance, making them well-suited for the SVHN dataset's varied backgrounds. While simpler models like LeNet-5 offer a balance between speed and accuracy, they might not be as effective in capturing the complexity present in SVHN images.