ASSIGNMENT_01

March 27, 2024

Question 01: 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. [3 Marks] [Theory]

Solution : The application of loss function depends on the nature of problem statement, how you want to deal with instance wise error you get while traning the model.In the logistic regression the output is probability vector representing the likelihood of instance belonging to each class termed as logits,obtained by passing the weighted sum of features through a sigmoid function.

Cross Entropy i.e. log loss measures the difference between the predicted probabilities and the actual labels. It calculates the loss as the negative log-likelihood of the labels given the model's predictions and . The loss is minimized when the predicted probabilities match the true labels.Cross Entropy loss ensures that the model aims to predict the probabilities as close to the true labels as possible. It penalizes the model more heavily for wrong predictions, thus pushing the model to learn better feature representations.

While in the case of Mean Squared Error loss it measures the squared difference between predicted values and true labels.It is not suitable for classification tasks as it doesn't directly optimize probabilities thus leading suboptimal results. We can conclude through this that cross entropy will be work better than Mean squared error loss while working with logistic regression as Cross Entropy loss ensures that logistic regression models produce probabilities that are as close as possible to the true labels.
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 [3 Marks] [Theory]

Solution : (option b).For a binary classification task with a deep neural network (containing at least one hidden layer) equipped with linear activation functions, using the Mean Squared Error (MSE) loss function is best suitable for guaranteeing convex optimization.Using linear activation functions, linearity gets introduced in hidden layers which ensures convex optimization.

The convexity of the MSE loss function, driven by its squared term, further aids the convex nature of the optimization problem, as differences between predictions from true labels , the errros are penalized in a convex manner. Which further ensures that any local minima encountered during optimization is also a global minimum.

Cross entropy loss might not guarantee convex optimixation as it involves logarithmic terms which introduces non-linearity in network, which may further leads formation of multiple local minima, making it challenging for optimization algorithms like gradient descent to converge to the global minimum, high chances getting trapped in local minima.Hence, Mean Squared Error loss is preferred for convex optimization problems assuring convergence to the global minimum, essential for the effective training of deep neural networks.

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. [2 for implementation and 3 for explanation] [Code and Report]

Solution : Code is mentioned in attached file- `question_03.py`

Data preprocessing and hyperparameter tunning for CNN on MNIST dataset.

Problem statement:

Developing CNN architecture and deploying it for image classification task on MNIST dataset along with application of data preprocessing and fine tunning hyperparameters.

Experimental setup:

Dataset details:

TMNIST dataset for handwritten digit recognition consisting of 60,000 training examples and 10,000 testing examples. Each example is a 28 $\times$ 28 size gray-level image, single channel.

Data preprocessing details:

1. **Normalization:** Pixel values of the input images were normalized to the range [0, 1].
2. **Reshaping:** Reshaping process is carried out on input image for including channel dimension, so that input image can be passed through CNN architecture.

Model details :

The CNN architecture comprised multiple convolutional and pooling layers followed by fully connected layers for classification. The architecture was defined as follows:

- Convolutional layer: 32 filters, kernel size (3, 3), ReLU activation
- MaxPooling layer: Pool size (2, 2)
- Convolutional layer: 64 filters, kernel size (3, 3), ReLU activation
- MaxPooling layer: Pool size (2, 2)
- Flatten layer
- Dense layer: 128 neurons, ReLU activation
- Output layer: 10 neurons (softmax activation)
- Here we used Flatten layer, Dense layer, and Output layer.

Hyperparameter tuning:

1. Hyperparameters tuned: Number of neurons in dense layer: 64,128,256
2. Activation function: Sigmoid,tanh,ReLU.

Grid search was employed using scikit-learn's `GridSearchCV` with 3-fold cross-validation. The grid search explored various combinations of neurons in the dense layer and activation functions. The model's performance was evaluated based on the mean validation accuracy across folds.

Results

- This model utilized ReLU activation with 256 neurons in the dense layer.
- ReLU activation consistently outperformed tanh and sigmoid activation functions across different numbers of neurons in the dense layer.
- Increasing the number of neurons generally improved performance, with the highest accuracy achieved using 256 neurons.
- The test accuracy of the best model on the unseen test set was 96.35999798774719%.

Conclusion

The application of data preprocessing techniques and hyperparameter tuning can definitely help us in improving the performance of applied model on MNISTdataset.

Recommendations

Based on the results, ReLU is best suited with 256 neurons in hidden layers on MNIST dataset.However there is still scope to employ other preprocessing and hyperparameter tuning techniques, architectural modifications can be further explored.

Question 04: Build a classifier for Street View House Numbers (SVHN) (Dataset) using pre-trained model weights from PyTorch. Try multiple models like LeNet-5, AlexNet, VGG, or ResNet(18, 50, 101). Compare performance comments on why a particular model is well suited for the SVHN dataset. (You can use a subset of the dataset (25%) in case you do not have enough compute.) [4 Marks] [Code and Report]

Solution : Code is mentioned in attached file- `question_04.py`

Performance comparison of pretrained models for given classification task on SVHN dataset.

Problem statement:

To compare the performance of various models using pretrained weights from PyTorch for classification task evaluated on given SVHN dataset.

Dataset details:

Street View House Numbers (SVHN) is real world, digit classification benchmark dataset that contains 600,000 32×32 RGB images of printed digits (from 0 to 9) cropped from pictures of house number plates.Due to computational constraint experiment is done only 25 % of the given dataset.

Model details:

LeNet-5, AlexNet, VGG, ResNet18, ResNet50, ResNet101

Evaluation Metrics:

The performance of these models on test data are evaluated against following metrics:

- Test Accuracy: TP+FN/TP+TN+FP+FN
- Precision: TP/TP+FP
- Recall: TP/TP+FN
- F1-score: 2* Precision* Recall/ Recall+ Precision

Results

Model	Test Accuracy	Precision	Recall	F1-score
LeNet-5	77.74%	0.769	0.751	0.757
VGG-16	83.19%	0.830	0.814	0.814
ResNet-18	88.58%	0.877	0.876	0.875
ResNet-50	87.12%	0.863	0.862	0.861
ResNet-101	79.61%	0.800	0.778	0.777

Analysis

While going through accuracy and other metrics on test dataset we can conclude that Resnet18 and Resnet 50 outperformed other models.Same can be explained by many reason out of which their architecture remains initial one of them.Along with deeper architecture and skip connections further facilitates model facilitates model for smooth gradient flow overcome vanishing gradient problem and enabling effective deep network training,facilitating better convergence.

But still , we can check that Resnet101 results lesser accuracy as compared to the other two resnet models probably the reason can be since this model have deeper architecture, the overall complexity increases which can further result in overfitting.

Lenet-5 due to its simpler architecture can learn limited complex features in comparison to other deeper models, although performed fairly well with accuracy of 77.74% on SVHN dataset.VGG has deeper architecture as compared Lenet but in absence of skip connections, its performance is lower compared to Resnet.

Conclusion

After checking evaluation metrics since Resnet18 has outperformed all other models, hence it is best suitable for the given problem statement.