Deep Learning: Assignmnet-1

Aditya Kishore 21011

March 27, 2024

Question 1 Answer:

Logistic Regression is a classification algorithm, and the best loss function to use with it is cross-entropy. It is preferred over Mean Squared error (MSE) because it guarantees a single best answer without ambiguity.This is crucial in training models as it ensures clear Optimization towards the correct classification in a proposed problem as Mean Squared error(MSE) is more suitable for regression tasks.

- Loss Functions: A loss function measures how much deviation occurred in the model's predictions from the actual values. The goal during training is to minimize this loss function and Logistic Regression is commonly used for binary classification tasks where the goal is to predict the probability that an input belongs to a particular class (e.g., positive or negative, spam or not spam, cat or not cat).
- Cross-Entropy for Classification: Logistic Regression outputs a Probability between 0 and 1 to represent the likelihood of a class.Cross-entropy penalizes the model for incorrect probability estimates. It goes to zero when the predicted probabilities perfectly match the actual labels (0 or 1 in binary classification).
- Mean Squared Error for Regression: It Calculates the average squared difference between predicted and actual values.It is better suited fore regression tasks where we predict continuous values, and minimizing the squared errors leads the model to get closer to the actual numerical values.

Impact on Training:

- Convexity: For logistic regression, cross-entropy results in a convex loss function. It is ideal that there is just one minimal(single minimum) point as a result.Where as combining MSE with logistic regression can result in a non-convex loss function with many minima which makes it more difficult to identify the best solution during training.

Question 2 Answer:

For a binary classification task with a deep neural network containing at least one hidden layer and linear activation functions, the loss function guarantees a convex optimization problem is Cross Entropy(CE).

- Formal Proof for Cross Entropy(CE) Loss Function:

  The Cross Entropy loss function for binary classification is defined as:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

  where: - y is the true label (0 or 1), - $\hat{y}$ is the predicted probability, - N is the number of samples.

Convexity Proof:

To show that the Cross Entropy loss function is convex, we need to demonstrate that its Hessian matrix is positive semi-definite.

The Hessian matrix of the Cross Entropy loss function is given by:

$$H_{ij} = \frac{\partial^2 L}{\partial w_i \partial w_j}$$

For the Cross Entropy loss function, the Hessian matrix is:

$$H = \frac{1}{N} X^T S X$$

where: - X is the input data matrix, - S is a diagonal matrix with elements $\hat{y}_i(1 - \hat{y}_i)$.

Since the Hessian matrix is a positive semi-definite matrix, the Cross Entropy loss function is convex.

Conclusion:

Therefore, the Cross Entropy loss function guarantees convex optimization for binary classification tasks with deep neural networks containing linear activation functions, ensuring a smooth and efficient training process without local minima issues.

Question 3 Answer:

To Implement a feedforward neural network with dense layers only, we can use Keras, and we need a Neural network architecture for it, Processing of input images and Hyperparameter Tuning Strategies, The Dataset I am using is MNIST IMAGE RECOGNITION DATASET:

Neural network Architecture:

- No. of Hidden Layers: I am using 2 hidden layers.
- Neurons Per layers: I am choosing 128 neurons for the first hidden layer and 64 neurons for the second hidden layer.(No. of Neurons depends upon the complexity of the problem and size of dataset).
- Activation Function: I am using Rectified Linear Activation Function (ReLU) for hidden layers and sigmoid function for the output layer since it more preferred for binary classification tasks

Preprocessing:

- Rescaling: Scaling the pixel values to a range between 0 and 1.
- Normalization: Normalizing the pixel values to have a mean of 0 and standard deviation of 1.
- Resizing: Resizing the images to a uniform size because there can be images that are varying dimensions.
- Data Augmentation: Applying random transformations to increase the diversity of the training dataset.

Hyperparameter Tuning Strategies:

- Grid Search: Searching through a grid of hyperparameters to finding out the best combination.
- Random Search: Randomly sampling hyperparameters from predefined ranges.
- Bayesian Optimization: Using Bayesian methods to optimizing hyperparameters efficiently.
- Cross-validation: Using cross-validation to evaluate the performance of different hyperparameter configurations. we can use cross-val-score or K-Fold cross validation techniques

The Model is using Adam Optimizer(optimization technique for gradient descent) to minimize the loss function during the training of neural networks.And for calculating loss Binary cross-entropy technique is adopted to measure the dissimilarity between the predicted probability distribution and the true binary labels of a dataset.The number of Epochs on which the model is trained is 100 and 10 to show variations in the metrics i.e Accuracy, considering the same batch size of 32 images for both of the number of epochs.

Metrics	**Results**
Loss	-5559442472960.00
Accuracy	0.1124
Val-loss	-5750832758784.00
Val-accuracy	0.1135

Table 1: Results for 100 Epochs

Metrics	**Results**
Loss	-10458409984.00
Accuracy	0.1124
Val-loss	-12135690240.00
Val-accuracy	0.1135

Table 2: Results for 10 Epochs

Question 4 Answer:

To meet the computational cost the number of epochs are restricted to 10 and 15(LeNet-5) as computational resources are limited.

LeNet-5:

It is consistently gaining accuracy over 15 epochs, from 18% to 84% which is remarkable for such an early and basic convolutional network But compared to deeper models, its comparatively simpler design could make it less able to capture the more complex elements in the SVHN dataset.

AlexNet:

AlexNet's performance stayed mostly constant at 19% accuracy during the training phase. This constant behaviour shows that even while the model is deeper than LeNet-5, it could not be as good at managing the SVHN dataset's complexity or variability, perhaps because of the way its architecture handles picture attributes.

VGG16:

For the SVHN dataset, VGG-16 had highly inconsistent performance with loss values spiking to extremely high amounts like from epoch 4 loss: 2.28 to epoch 5 loss: 5421.81. This suggests that there may be problems with overfitting, learning rate settings, or data preparation inconsistencies. Without some tweaking, its underlying architecture—which is renowned for its effectiveness in large-scale picture recognition—might not translate well to the unique characterstics of SVHN.

ResNet-18:

With a remarkable 81% accuracy in the first epoch and a consistent rise to 91% by the tenth epoch, ResNet-18 stands out. It is ideally suited for this purpose because residual connections let it to learn from the SVHN dataset without performance being negatively impacted by the vanishing gradient issue.

ResNet-50 and ResNet-101:

Although both models exhibit notable progress over time, they are not as good as ResNet-18. This may be the result of these models' greater complexity, which may make it harder to train on the very small sample of the SVHN dataset that was employed. This suggests that, for this particular job, additional layers may not be as beneficial after a certain depth.

Models	Epoch 1	Epoch 10
ResNet-50	19%	87%
ResNet-101	19%	86%

Table 3: Performance of the ResNet-50 and ResNet-101 over 10 Epochs.

Refrences:

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012

2. Y. LeCun, C. Cortes, C. Burges, et al. Mnist handwritten digit database, 2010.