An investigation of classification method (logistic regression) for CIFAR-10

**Dataset**

The CIFAR-10 dataset is the dataset of real-color images is frequently used to gauge a deep neural network's capacity for detection. 60000 tiny 30x30 pixel photos from 10 classes (categories) are included in this dataset

Categories include plane, car, bird, cat, deer, dog, frog, horse, ship, truck. By default, the dataset is divided into two components: Training set contain 50000 images while set for testing include 10000 images i.e. 1000 images separated for each category. Images are captured under a range of circumstances, such as position, size, pose, and illumination. Numerous items have partial occlusion. The precise values for that is also used in pytorch for CIFAR-10 train set over the three channels (r,g,b). The mean values: **0.49139968**, **0.48215827** ,**0.44653124** and Standard deviation values: **0.24703233,** **0.24348505**, **0.26158768.**

|  |
| --- |
| Figure : CIFAR-10 Dataset |

**Logistics regression**

The Supervised Learning method includes the Machine Learning algorithm of logistic regression. Using a predetermined set of independent factors, it is used to predict the categorical attributes. Predictive analytics and classification are frequent uses of it. Additionally, it is thought of as a discriminative model, which implies it makes an effort to distinguish between different classes (or categories). The most useful case of Logistics regression is classification. Classification is called as binary(0/1) problems, where the output is either 0 or 1, Yes or No. If the classes are more than two then this type of logistics regression become Multi class classification

This experiment is carried out using the PyTorch package. Learning Rate (α) is the hyperparameter that was tweaked during the experiment. How quickly we update the settings depends on the learning rate. We might "overshoot" the ideal value if the learning rate is too high. In a similar manner, if it is too tiny, it will take too long for us to reach the ideal values. Consequently, it is essential to employ a calibrated learning rate. In this experiment we use stochastic gradient descent and Batch Gradient Descent for optimization

**Experimental Methodology**

* Stage 01 ()
  + Explore the dataset i.e. using basic exploratory data analysis (EDA) we glean that The CIFAR-10 data set provides 10 classes of pictures, with training data sets totaling 50000 images and test data sets totaling 10,000, according to basic EDA. Every picture measures [3 x 32 x 32]. This denotes a 32 × 32 pixel image with three RGB channels.
  + The relevant libraries, including pandas, numpy, matplotlib, torch, and torchvision, are being imported
* Stage 02 (Prepare training data.)
  + In this stage we split data in 3 sets i.e. training, validation, testing in the percentage of 60,20,20 respectively. Through training set we train our model, we adjust the weights for accuracy and to compute losses. The validation set is used to appraise the hyperparameter of model & choose the finest model while being trained. The test data set is Used to evaluate the final correctness of several models and compare them in the context of accuracy
* Stage 03 (Developing a foundation model class and GPU training)
  + In this stage we create the class ImageClassificationBase that we are creating derives from Logistic Regression (torch.nn.Module). This does not include the \_\_init\_\_ and \_\_forward\_\_ functions from the model architecture
* Stage 04 (Model of Training)
  + In this stage two hidden layers have been employed. 2048 x 1650 x 512 x 138 x 10 will be the neural network architecture. Flattening images into vectors, applying layers, activation functions, and receiving predictions using the output layer are the last steps. Here, the activation function Relu is utilised.
  + Fit function is used to train the model which help to increase accuracy and decrease loss. Here, we experiment with various learning rates (which is the hyperparameter) and epochs. For the highest accuracy, we choose epochs 25 and a learning rate of 0.001.

**Hyperparameter & Metric**

I focused on Accuracy metric with respect to loss value in order to get the best performance of model. As we can see from the figures below that the tuning of hyperparameter is at its best the epochs vs loss is saturated, as we can see the number of epoch is 77 loss value is 1.391 and in the next figure the number of epoch are 81 where there is no change in loss and the value of loss almost remain 1.391 we can also visualize this result as well. Vice versa the same happened with accuracy vs no of epochs where in the first image the number of epochs are 77 and accuracy is 0.508 and in the next image the number of epochs are 81 and accuracy almost remain same that is 0.508

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

**Result**

Viewing the accuracy score relative to the labels in the training set is a logical way to assess the model. We focused on accuracy metric with respect to loss to evaluate the model performance The computing results show that for SGD (Stochastic Gradient Descent) and BGD (Batch Gradient Descent), both training accuracy and validation accuracy can exceed 50%. The SGD's loss reducing (stochastic gradient descent) is substantially faster than BGD's (Batch Gradient Descent). BGD (Batch Gradient Descent) requires more time to converge depending on the chosen learning rate strength. The learning rate cannot be easily determined.. Firstly I use large learning rate, and the loss goes up then observing the epochs with respect to loss and accuracy I decrease the learning rate and at the end I get fine result at the learning rate of 0.001.On test data the accuracy we achieved is 0.51 which 51%

|  |
| --- |
|  |