Experiment No. 3
Apply any of the following learning algorithms to learn the parameters of the supervised single layer feed forward neural network. a. Stochastic Gradient Descent b.Mini Batch Gradient Descent c.Momentum GD d.Nestorev GD e.Adagrad GD f. Adam Learning GD
Date of Performance: 08/02/2024
Date of Submission: 15/02/2024

Title: Apply any of the following learning algorithms to learn the parameters of the supervised single layer feed forward neural network. a. Stochastic Gradient Descent b. Mini Batch Gradient Descent c. Momentum GD d. Nestorev GD e. Adagrad GD f. Adam Learning GD

Aim: Apply learning algorithms to learn the parameters of the supervised single layer feed forward neural network using Stochastic Gradient Descent.

Objective: To implement various training algorithms for feedforward neural networks.

Theory:

Gradient descent is an optimization algorithm which is commonly-used to train machine learning models and neural networks. Training data helps these models learn over time, and the cost function within gradient descent specifically acts as a barometer, gauging its accuracy with each iteration of parameter updates. These factors determine the partial derivative calculations of future iterations, allowing it to gradually arrive at the local or global minimum (i.e. point of convergence).

Learning rate (also referred to as step size or the alpha) is the size of the steps that are taken to reach the minimum. This is typically a small value, and it is evaluated and updated based on the behaviour of the cost function.

The cost (or loss) function measures the difference, or error, between actual y and predicted y at its current position. This improves the machine learning model's efficacy by providing feedback to the model so that it can adjust the parameters to minimize the error and find the local or global minimum.

Types of gradient descent

There are three types of gradient descent learning algorithms: batch gradient descent, stochastic gradient descent and mini-batch gradient descent.

- Batch gradient descent

Batch gradient descent sums the error for each point in a training set, updating the model only after all training examples have been evaluated. This process referred to as a training epoch.

- Stochastic gradient descent

Stochastic gradient descent (SGD) runs a training epoch for each example within the dataset and it updates each training example's parameters one at a time.

- Mini-batch gradient descent

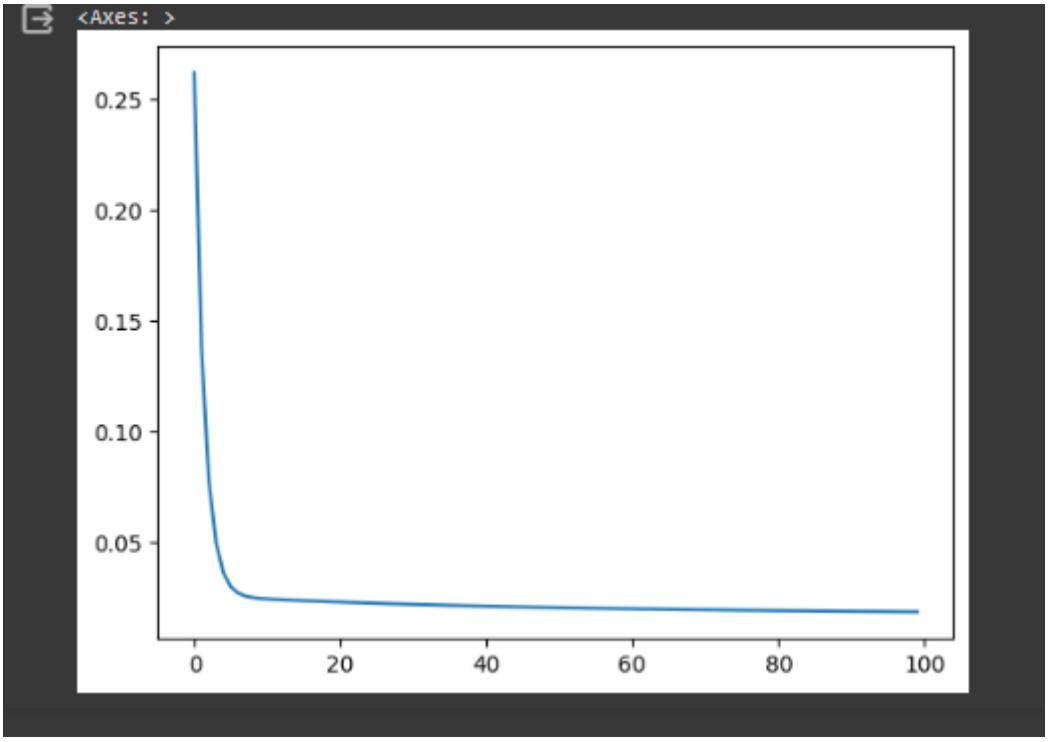
Mini-batch gradient descent combines concepts from both batch gradient descent and stochastic gradient descent. It splits the training dataset into small batch sizes and performs updates on each of those batches.

Code :

```
 import pandas as pd
from IPython.display import display
red_wine=pd.read_csv('red-wine.csv')
df_train=red_wine.sample(frac=0.7,random_state=0)
df_valid=red_wine.drop(df_train.index)
display(df_train.head(4))

#scale 0-1
max_=df_train.max(axis=0)
min_=df_train.min(axis=0)
df_train=(df_train-min_)/(max_-min_)
df_valid=(df_valid-min_)/(max_-min_)



#split f and t
X_train = df_train.drop('quality', axis=1)
#print(X_train)
X_valid = df_valid.drop('quality', axis=1)
#print(X_valid)
y_train = df_train['quality']
#print(y_train)
y_valid = df_valid['quality']
#print(y_valid)
print(X_train.shape)


from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(512, activation='relu', input_shape=[11]),
    layers.Dense(512, activation='relu'),
    layers.Dense(512, activation='relu'),
    layers.Dense(1),
])


model.compile(
    optimizer='sgd',
    loss='mse',
)


history = model.fit(
    X_train, y_train,
    validation_data=(X_valid, y_valid),
```

```
    batch_size=256,
    epochs=100
)


history_df=pd.DataFrame(history.history)
history_df['loss'].plot()
```

Output :



Conclusion: Stochastic Gradient Descent (SGD) is a powerful optimization algorithm commonly used in machine learning. It operates by iteratively updating the model parameters based on the gradient of the loss function computed on small random subsets of the training data. Despite its simplicity, SGD exhibits fast convergence and scalability to large datasets. However, its stochastic nature introduces variability in the parameter updates, which can result in noisy convergence trajectories. Regularization techniques and careful tuning of hyperparameters are often necessary to ensure optimal performance. Overall, SGD serves as a cornerstone in training deep learning models, offering a balance between computational efficiency and effectiveness in finding optimal solutions.