NAME : PAREENITA SHIRSATH PRN : 221101062 B.E.A.I.&.D.S.

DLL EXPERIMENT NO : 04

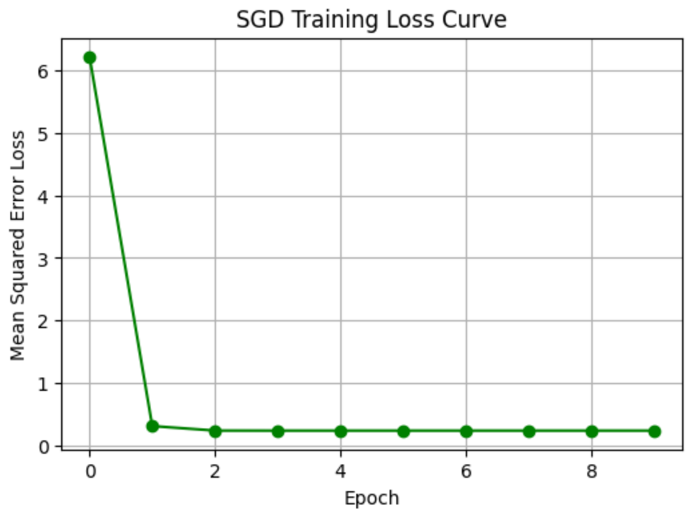
AIM : Apply any of the following learning algorithms to learn the parameters of the supervised single layer feedforward neural network: a. Stochastic Gradient Descent, b. Mini Batch Gradient Descent, c. Momentum GD, d. Nesterov GD, e. Adagrad GD, f. Adam Learning GD

a. Stochastic Gradient Descent (SGD)

Explanation: Updates are made after each training example, giving very frequent updates. May be noisy but fast to converge in some cases.

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import models, layers, optimizers

# Set seeds for reproducibility
np.random.seed(0)
tf.random.set_seed(0)

# Generate dummy data
X = np.random.randn(100, 3)  # 100 samples, 3 features
true_W = np.array([[2.0], [-3.5], [1.0]])
y = X @ true_W + 0.5 * np.random.randn(100, 1)  # add noise

# Build a simple linear regression model with Keras
model = models.Sequential([
    layers.Dense(1, input_shape=(3,), use_bias=True)
])

# Compile the model with SGD optimizer
model.compile(
    optimizer=optimizers.SGD(learning_rate=0.01),
    loss='mse'
)

# Train the model with batch_size=1 to mimic true SGD
history = model.fit(
    X, y,
    epochs=10,
    batch_size=1,
    verbose=1
)

# Extract learned weights and bias
weights, bias = model.layers[0].get_weights()

print("Learned weights:", weights.flatten())
print("Learned bias:", bias[0])

# Plot loss curve
plt.figure(figsize=(6, 4))
plt.plot(history.history['loss'], marker='o', color='green')
plt.title('SGD Training Loss Curve')
plt.xlabel('Epoch')
plt.ylabel('Mean Squared Error Loss')
plt.grid(True)
plt.show()
```

```
Epoch 1/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 11.2055
Epoch 2/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.3511
Epoch 3/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.2340
Epoch 4/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2319
Epoch 5/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2316
Epoch 6/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2315
Epoch 7/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2315
Epoch 8/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2315
Epoch 9/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2315
Epoch 10/10
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2315
Learned weights: [ 2.0028615 -3.5358438  1.0312603]
Learned bias: -0.08746299
```



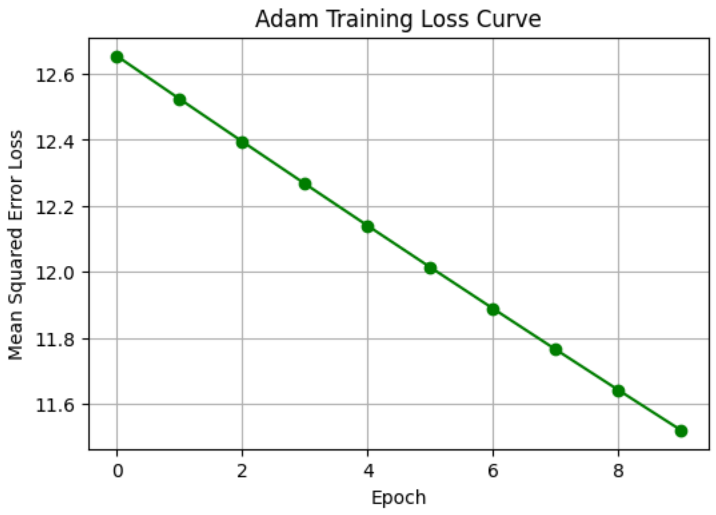
f. Adam Optimizer

Explanation: Combines momentum and adaptive learning rate. One of the most popular optimizers in deep learning.

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import models, layers, optimizers

# Set random seeds for reproducibility
np.random.seed(0)
tf.random.set_seed(0)

# Generate dummy data
X = np.random.randn(100, 3)  # 100 samples, 3 features
true_W = np.array([[2.0], [-3.5], [1.0]])
y = X @ true_W + 0.5 * np.random.randn(100, 1)  # add noise

# Build a simple linear regression model with Keras
model = models.Sequential([
    layers.Dense(1, input_shape=(3,), use_bias=True)
])

# Compile the model with Adam optimizer
model.compile(
    optimizer=optimizers.Adam(learning_rate=0.01),
    loss='mse'
)

# Train the model with full batch
history = model.fit(
    X, y,
    epochs=10,
    batch_size=100,
    verbose=1
)

# Extract learned weights and bias
```

```
weights, bias = model.layers[0].get_weights()

print("Learned weights:", weights.flatten())
print("Learned bias:", bias[0])

# Plot loss curve
plt.figure(figsize=(6, 4))
plt.plot(history.history['loss'], marker='o', color='green')
plt.title('Adam Training Loss Curve')
plt.xlabel('Epoch')
plt.ylabel('Mean Squared Error Loss')
plt.grid(True)
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 461ms/step - loss: 12.6539
Epoch 2/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 12.5241
Epoch 3/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - loss: 12.3952
Epoch 4/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - loss: 12.2673
Epoch 5/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - loss: 12.1404
Epoch 6/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 153ms/step - loss: 12.0145
Epoch 7/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step - loss: 11.8896
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step - loss: 11.7658
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 11.6430
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - loss: 11.5212
Learned weights: [ 0.70711994 -0.8019474  -0.12943842]
Learned bias: -0.09887612
```


Adam Training Loss Curve
Mean Squared Error Loss
Epoch