Step 1: Import libraries

```
# Import libraries

import numpy as np
import pandas as pd
import matplotlib.cm as cm
import matplotlib.pyplot as plt

import plotly.express as px
import plotly.graph_objs as go

from tensorflow import keras
from keras.layers import Dense
from keras.models import Sequential
from keras.optimizers import SGD

#import warnings
#warnings.filterwarnings('ignore')
```

```
pip install -U kaleido
```

```
Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 9.1 MB/s eta 0:00:00
Installing collected packages: kaleido
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
lida 0.0.10 requires fastapi, which is not installed.
lida 0.0.10 requires python-multipart, which is not installed.
lida 0.0.10 requires uvicorn, which is not installed.
Successfully installed kaleido-0.2.1
```

Step2: Take XOR input data and store in one variable (Input data), Store output data of XOR in another variable (Target data)

```
X = np.array([0,0,1,1]).reshape(4, 1)
Y = np.array([0,1,0,1]).reshape(4, 1)
```

```
input_data = np.hstack((X, Y))
input_data
```

```
array([[0, 0],
       [0, 1],
       [1, 0],
       [1, 1]])
```

```
target_data = np.array([0,1,1,0])
```

Step3: Create the model.

Define sequential model.

```
model = Sequential()
```

Add first layer in the model

```
model.add(Dense(8, input_shape=(2,), activation='relu'))
```

Add the second layer in the model

```
model.add(Dense(1, activation='sigmoid'))
```

Keep learning rate = 0.1

```
LEARNING_RATE = 0.1
```

Use SGD as an optimizer with given learning rate.

```
sgd = SGD(learning_rate = LEARNING_RATE)
```

Step4: Compile the model with the defined optimizer in the previous step with MSE as the loss term.

```
model.compile(loss='mean_squared_error', optimizer = sgd)
```

TASK 1: Convergence speed for the default case

Step5: Now, we need to record the learning rates so that we can capture the number of epochs at model converging.

```
TAREGT_LOSS = 0.002
```

```
# Custom Callback

learning_rates = []
class LearningRateCallback(keras.callbacks.Callback):
    '''Appends learning rate every epoch into a list'''

    def on_epoch_begin(self, epoch, logs=None):
        lr = self.model.optimizer.lr.numpy()
        learning_rates.append(lr)
```

Step6: Train the model and monitor the convergence and learning rates.

```
epochs = 0
mse_losses = []

while True:
    # Train the model for 10 epochs
    history = model.fit(input_data, target_data, epochs = 10, verbose = 1, callbacks=[LearningRateCallback()])

    epochs += 10
    training_loss_values = history.history['loss']    # Training loss

    mse_losses.extend(training_loss_values)
    previous_loss = training_loss_values[-1]

    # Check convergence
    if previous_loss <= TAREGT_LOSS:
        print(f'Model converged after {epochs} epochs.')
        break
```

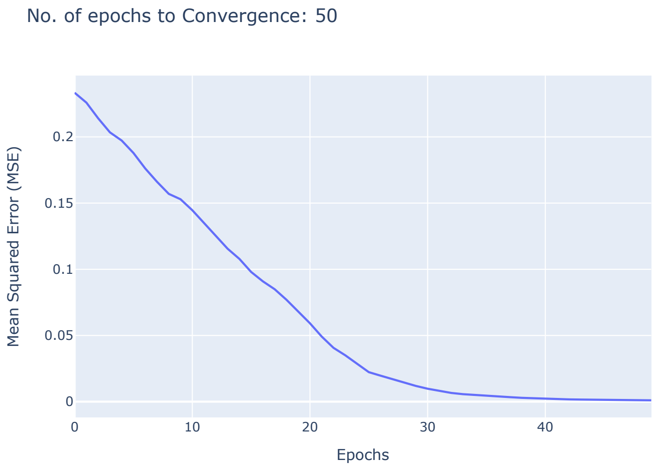
```
Epoch 10/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0020
Model converged after 3020 epochs.
```

```
print(f'Model converged after {epochs} epochs.')
```

```
Model converged after 3020 epochs.
```

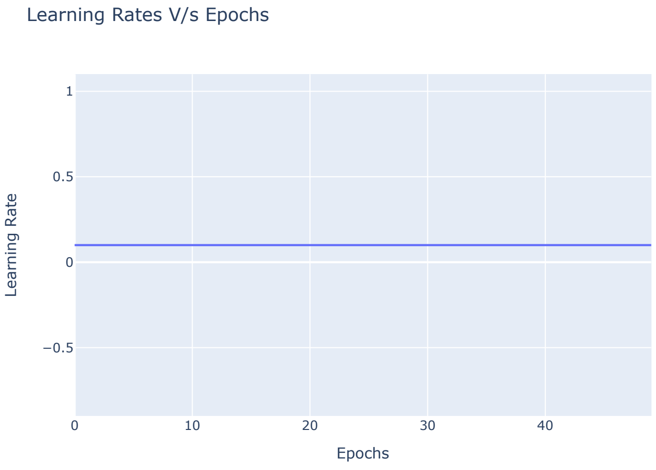
Step7: Plot the SSE (Sum of squared error) vs. Number of epochs.

```
x_axis_str = 'Epochs'
y_axis_str = 'Mean Squared Error (MSE)'

fig = px.line(x = range(epochs), y = mse_losses,
              width = 700,
              height = 500,
              title = f'No. of epochs to Convergence: {epochs}',
              labels = {'x' : x_axis_str, 'y' : y_axis_str})
fig.show()
fig.write_image('No. of epochs to Convergence - Default case.png')
```



Step8: Plot the graph of learning rate vs Number of epochs.

```
x_axis_str = 'Epochs'
y_axis_str = 'Learning Rate'

fig = px.line(x = range(epochs), y = learning_rates,
              width = 700,
              height = 500,
              title = 'Learning Rates V/s Epochs',
              labels = {'x' : x_axis_str, 'y' : y_axis_str})
fig.show()
fig.write_image('Learning Rates Vs Epochs - Default case.png')
```



TASK 2: Understanding the effect of momentum on convergence speed

Step9: Accelerate learning by incorporating momentum based learning rate.

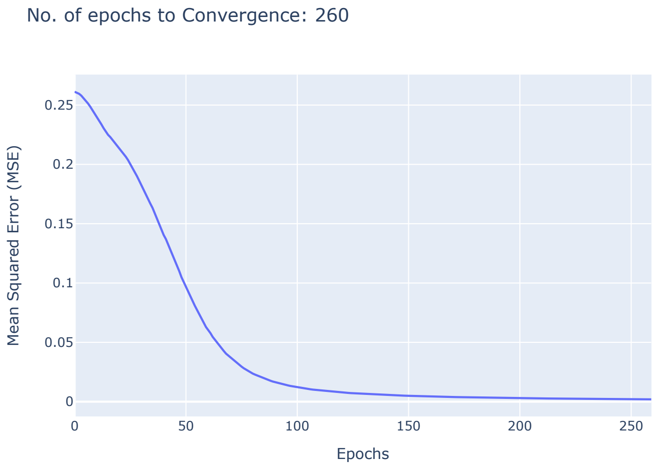
```
MOMENTUM = 0.9


model2 = Sequential()
model2.add(Dense(8, input_shape=(2,), activation='relu'))
model2.add(Dense(1, activation='sigmoid'))

sgd = SGD(learning_rate = LEARNING_RATE, momentum = MOMENTUM)
model2.compile(loss='mean_squared_error', optimizer = sgd)

epochs = 0
mse_losses = []
learning_rates = []

while True:
    # Train the model for 10 epochs
    history = model2.fit(input_data, target_data, epochs = 10, verbose = 1, callbacks=[LearningRateCallback()])

    epochs += 10
    training_loss_values = history.history['loss']    # Training loss

    mse_losses.extend(training_loss_values)
    previous_loss = training_loss_values[-1]

    # Check convergence
    if previous_loss <= TAREGT_LOSS:
        break
```

```
Epoch 9/10
1/1 [==============================] - 0s 6ms/step - loss: 0.0058
Epoch 10/10
1/1 [==============================] - 0s 6ms/step - loss: 0.0057
Epoch 1/10
1/1 [==============================] - 0s 17ms/step - loss: 0.0057
Epoch 2/10
1/1 [==============================] - 0s 9ms/step - loss: 0.0056
Epoch 3/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0055
Epoch 4/10
1/1 [==============================] - 0s 9ms/step - loss: 0.0054
Epoch 5/10
1/1 [==============================] - 0s 10ms/step - loss: 0.0053
Epoch 6/10
1/1 [==============================] - 0s 10ms/step - loss: 0.0053
Epoch 7/10
1/1 [==============================] - 0s 9ms/step - loss: 0.0052
Epoch 8/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0051
Epoch 9/10
1/1 [==============================] - 0s 10ms/step - loss: 0.0051
Epoch 10/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0050
Epoch 1/10
1/1 [==============================] - 0s 15ms/step - loss: 0.0049
Epoch 2/10
1/1 [==============================] - 0s 13ms/step - loss: 0.0049
Epoch 3/10
1/1 [==============================] - 0s 11ms/step - loss: 0.0048
Epoch 4/10
1/1 [==============================] - 0s 11ms/step - loss: 0.0048
Epoch 5/10
1/1 [==============================] - 0s 10ms/step - loss: 0.0047
Epoch 6/10
1/1 [==============================] - 0s 10ms/step - loss: 0.0046
Epoch 7/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0046
Epoch 8/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0045
Epoch 9/10
1/1 [==============================] - 0s 6ms/step - loss: 0.0045
Epoch 10/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0044
Epoch 1/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0044
Epoch 2/10
1/1 [==============================] - 0s 5ms/step - loss: 0.0043
Epoch 3/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0043
Epoch 4/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0042
Epoch 5/10
1/1 [==============================] - 0s 12ms/step - loss: 0.0042
Epoch 6/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0041
Epoch 7/10
1/1 [==============================] - 0s 11ms/step - loss: 0.0041
```

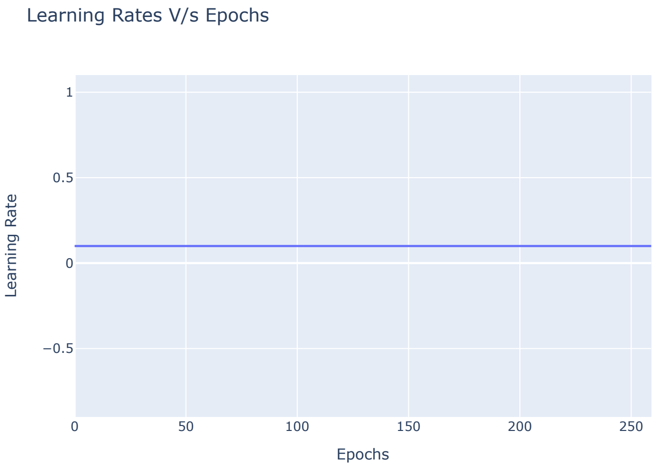
```
print(f'Model converged after {epochs} epochs.')
```

```
Model converged after 260 epochs.
```

```
x_axis_str = 'Epochs'
y_axis_str = 'Mean Squared Error (MSE)'

fig = px.line(x = range(epochs), y = mse_losses,
              width = 700,
              height = 500,
              title = f'No. of epochs to Convergence: {epochs}',
              labels = {'x' : x_axis_str, 'y' : y_axis_str})
fig.show()
fig.write_image('No. of epochs to Convergence - With Momentum.png')
```



```
x_axis_str = 'Epochs'
y_axis_str = 'Learning Rate'

fig = px.line(x = range(epochs), y = learning_rates,
              width = 700,
              height = 500,
              title = 'Learning Rates V/s Epochs',
              labels = {'x' : x_axis_str, 'y' : y_axis_str})
fig.show()
fig.write_image('Learning Rates Vs Epochs - With Momentum.png')
```



TASK 3: Understanding the effect of adaptive learning rate on convergence speed

Step10: Use adaptive learning rate.

```
from keras.optimizers import Adam

model3 = Sequential()
model3.add(Dense(8, input_shape=(2,), activation='relu'))
model3.add(Dense(1, activation='sigmoid'))

adam_optimizer = Adam(learning_rate = LEARNING_RATE)
model3.compile(loss='mean_squared_error', optimizer = adam_optimizer)

epochs = 0
mse_losses = []
learning_rates = []

while True:
    # Train the model for 10 epochs
    history = model3.fit(input_data, target_data, epochs = 10, verbose = 1, callbacks=[LearningRateCallback()])

    epochs += 10
    training_loss_values = history.history['loss']    # Training loss

    mse_losses.extend(training_loss_values)
    previous_loss = training_loss_values[-1]

    # Check convergence
    if previous_loss <= TAREGT_LOSS:
        break
```

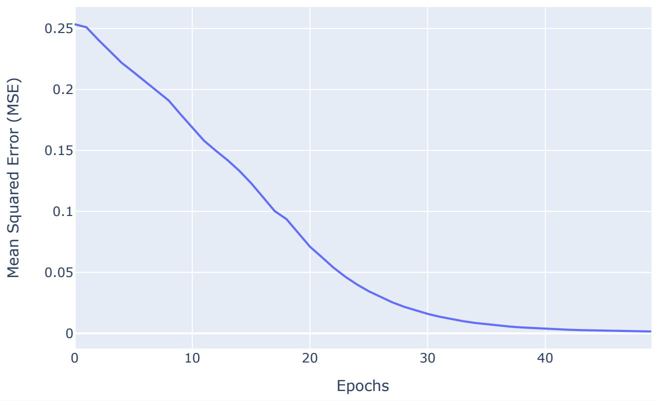
```
Epoch 1/10
1/1 [==============================] - 0s 500ms/step - loss: 0.2535
Epoch 2/10
1/1 [==============================] - 0s 7ms/step - loss: 0.2510
Epoch 3/10
1/1 [==============================] - 0s 7ms/step - loss: 0.2409
Epoch 4/10
1/1 [==============================] - 0s 6ms/step - loss: 0.2311
Epoch 5/10
1/1 [==============================] - 0s 6ms/step - loss: 0.2218
Epoch 6/10
1/1 [==============================] - 0s 7ms/step - loss: 0.2143
Epoch 7/10
1/1 [==============================] - 0s 6ms/step - loss: 0.2065
Epoch 8/10
1/1 [==============================] - 0s 5ms/step - loss: 0.1990
Epoch 9/10
1/1 [==============================] - 0s 6ms/step - loss: 0.1909
Epoch 10/10
1/1 [==============================] - 0s 9ms/step - loss: 0.1796
Epoch 1/10
1/1 [==============================] - 0s 15ms/step - loss: 0.1691
Epoch 2/10
1/1 [==============================] - 0s 7ms/step - loss: 0.1580
Epoch 3/10
1/1 [==============================] - 0s 11ms/step - loss: 0.1498
Epoch 4/10
1/1 [==============================] - 0s 9ms/step - loss: 0.1420
Epoch 5/10
1/1 [==============================] - 0s 9ms/step - loss: 0.1332
Epoch 6/10
1/1 [==============================] - 0s 6ms/step - loss: 0.1232
Epoch 7/10
1/1 [==============================] - 0s 7ms/step - loss: 0.1118
Epoch 8/10
1/1 [==============================] - 0s 6ms/step - loss: 0.1003
Epoch 9/10
1/1 [==============================] - 0s 6ms/step - loss: 0.0937
Epoch 10/10
1/1 [==============================] - 0s 5ms/step - loss: 0.0827
Epoch 1/10
1/1 [==============================] - 0s 11ms/step - loss: 0.0711
Epoch 2/10
1/1 [==============================] - 0s 10ms/step - loss: 0.0626
Epoch 3/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0539
Epoch 4/10
1/1 [==============================] - 0s 6ms/step - loss: 0.0465
Epoch 5/10
1/1 [==============================] - 0s 6ms/step - loss: 0.0401
Epoch 6/10
1/1 [==============================] - 0s 6ms/step - loss: 0.0345
Epoch 7/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0297
Epoch 8/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0254
Epoch 9/10
1/1 [==============================] - 0s 6ms/step - loss: 0.0218
```

```
print(f'Model converged after {epochs} epochs.')
```

```
Model converged after 50 epochs.
```

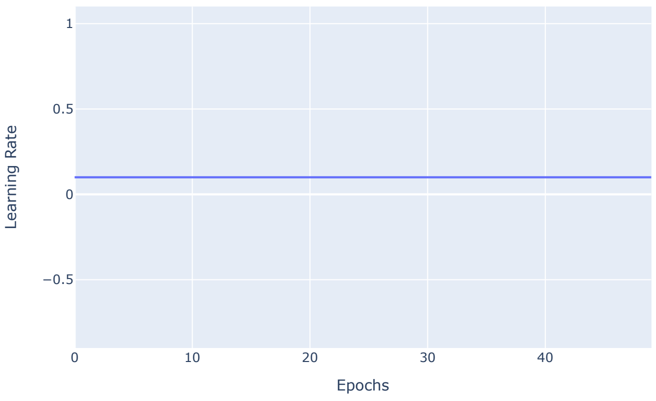
```
x_axis_str = 'Epochs'
y_axis_str = 'Mean Squared Error (MSE)'

fig = px.line(x = range(epochs), y = mse_losses,
              width = 700,
              height = 500,
              title = f'No. of epochs to Convergence: {epochs}',
              labels = {'x' : x_axis_str, 'y' : y_axis_str})
fig.show()
fig.write_image('No. of epochs to Convergence - Adam Optimizer.png')
```

No. of epochs to Convergence: 50



```
x_axis_str = 'Epochs'
y_axis_str = 'Learning Rate'

fig = px.line(x = range(epochs), y = learning_rates,
              width = 700,
              height = 500,
              title = 'Learning Rates V/s Epochs',
              labels = {'x' : x_axis_str, 'y' : y_axis_str})
fig.show()
fig.write_image('Learning Rates Vs Epochs - Adam Optimizer.png')
```

Learning Rates V/s Epochs



TASK 4: Using both momentum and adaptive learning rates for faster convergence

Step 11: Now use both Momentum and adaptive based learning rate.

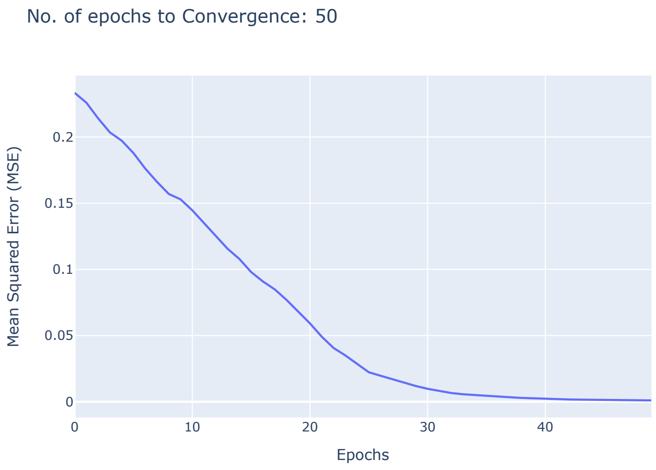
```
MOMENTUM = 0.9


model4 = Sequential()
model4.add(Dense(8, input_shape=(2,), activation='relu'))
model4.add(Dense(1, activation='sigmoid'))

adam_optimizer = Adam(learning_rate = LEARNING_RATE, beta_1 = MOMENTUM)
model4.compile(loss='mean_squared_error', optimizer = adam_optimizer)

epochs = 0
mse_losses = []
learning_rates = []

while True:
    # Train the model for 10 epochs
    history = model4.fit(input_data, target_data, epochs = 10, verbose = 1, callbacks=[LearningRateCallback()])

    epochs += 10
    training_loss_values = history.history['loss']    # Training loss

    mse_losses.extend(training_loss_values)
    previous_loss = training_loss_values[-1]

    # Check convergence
    if previous_loss <= TAREGT_LOSS:
        break
```

```
Epoch 2/10
1/1 [==============================] - 0s 5ms/step - loss: 0.0491
Epoch 3/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0406
Epoch 4/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0350
Epoch 5/10
1/1 [==============================] - 0s 5ms/step - loss: 0.0284
Epoch 6/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0223
Epoch 7/10
1/1 [==============================] - 0s 4ms/step - loss: 0.0196
Epoch 8/10
1/1 [==============================] - 0s 5ms/step - loss: 0.0170
Epoch 9/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0141
Epoch 10/10
1/1 [==============================] - 0s 6ms/step - loss: 0.0119
Epoch 1/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0098
Epoch 2/10
1/1 [==============================] - 0s 12ms/step - loss: 0.0079
Epoch 3/10
1/1 [==============================] - 0s 13ms/step - loss: 0.0066
Epoch 4/10
1/1 [==============================] - 0s 9ms/step - loss: 0.0057
Epoch 5/10
1/1 [==============================] - 0s 15ms/step - loss: 0.0050
Epoch 6/10
1/1 [==============================] - 0s 11ms/step - loss: 0.0044
Epoch 7/10
1/1 [==============================] - 0s 13ms/step - loss: 0.0039
Epoch 8/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0034
Epoch 9/10
1/1 [==============================] - 0s 11ms/step - loss: 0.0029
Epoch 10/10
1/1 [==============================] - 0s 13ms/step - loss: 0.0025
Epoch 1/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0021
Epoch 2/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0019
Epoch 3/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0017
Epoch 4/10
1/1 [==============================] - 0s 5ms/step - loss: 0.0016
Epoch 5/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0015
Epoch 6/10
1/1 [==============================] - 0s 9ms/step - loss: 0.0014
Epoch 7/10
1/1 [==============================] - 0s 7ms/step - loss: 0.0013
Epoch 8/10
1/1 [==============================] - 0s 8ms/step - loss: 0.0012
Epoch 9/10
1/1 [==============================] - 0s 9ms/step - loss: 0.0011
Epoch 10/10
1/1 [==============================] - 0s 9ms/step - loss: 0.0010
```

```
print(f'Model converged after {epochs} epochs.')
```

```
Model converged after 50 epochs.
```

```
x_axis_str = 'Epochs'
y_axis_str = 'Mean Squared Error (MSE)'

fig = px.line(x = range(epochs), y = mse_losses,
              width = 700,
              height = 500,
              title = f'No. of epochs to Convergence: {epochs}',
              labels = {'x' : x_axis_str, 'y' : y_axis_str})
fig.show()
fig.write_image('No. of epochs to Convergence - Adam Optimizer with Momentum.png')
```



```
x_axis_str = 'Epochs'
y_axis_str = 'Learning Rate'

fig = px.line(x = range(epochs), y = learning_rates,
              width = 700,
              height = 500,
              title = 'Learning Rates V/s Epochs',
              labels = {'x' : x_axis_str, 'y' : y_axis_str})
fig.show()
fig.write_image('Learning Rates Vs Epochs - Adam Optimizer with Momentum.png')
```