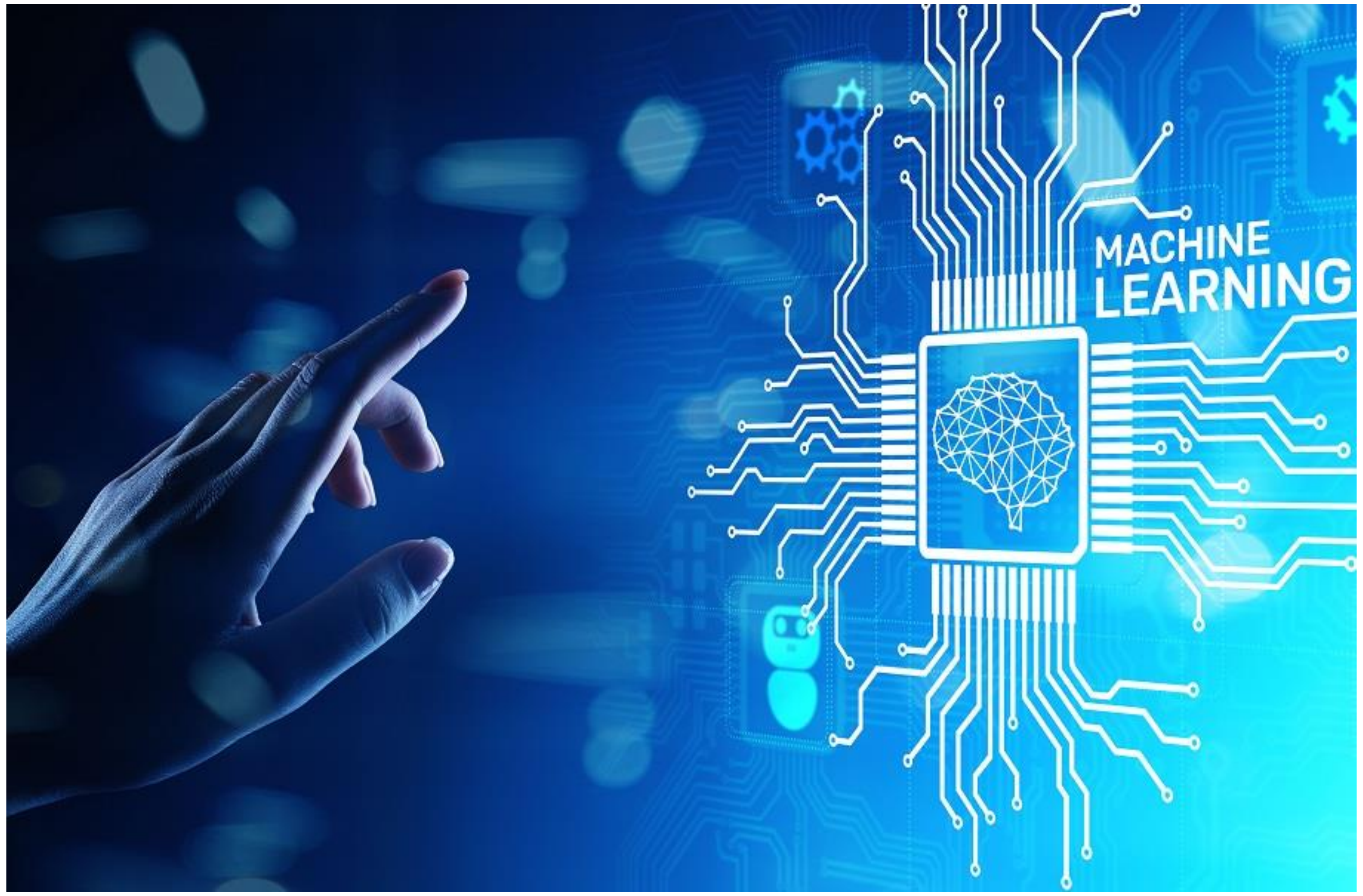
Duration: 100 min | BCSE424P | ML for Robotics Lab | Dr. Harini S

SLP, MLP



Learning for Today (10.02.2026)

- SLP

- MLP

SLP – Sample 1

```
import numpy as np

X = /* give 2 inputs for AND Gate */
y = /* Give true output for AND Gate */
w = np.zeros(2)
b = 0
eta = 0.1

for epoch in range(10):
        total_error = 0
        for i in range(len(X)):
                z = np.dot(X[i], w) + b
                y_pred = 1 if z >= 0 else 0
                error = y[i] - y_pred

                w = w + eta * error * X[i]
                b = b + eta * error

                total_error += abs(error)
                print("Weights:", w)
                print("Bias:", b)
          errors_per_epoch.append(total_error)
```

```
plt.plot(range(1, epochs+1),
errors_per_epoch)
plt.xlabel("Epochs")
plt.ylabel("Total Error")
plt.title("Learning Curve of Single Layer
Perceptron")
plt.show()
```

SLP – Sample 2 (Plot Decision Boundary)

```
/* Give inputs and outputs */
/* from training onwards use the below code */
slp = Perceptron(max_iter=1000, eta0=0.1,
random_state=1)
slp.fit(X, y)

x_min, x_max = -0.5, 1.5
y_min, y_max = -0.5, 1.5
xx, yy = np.meshgrid(np.linspace(x_min, x_max,
300),
np.linspace(y_min, y_max, 300))
# Predict over grid
Z = slp.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.3)
plt.scatter(X[:, 0], X[:, 1])
plt.xlabel("x1")
plt.ylabel("x2")
plt.title("Decision Boundary of SLP
for OR Problem")
plt.show()
```

Exercises

1. Implement SLP for OR GATE
2. Implement SLP for NOT Gate

Plot the error over epochs (Learning curve) and decision boundary for both the above questions

Sample 3: MLP for XOR (Binary Classifier}

```
from sklearn.neural_network import MLPClassifier
import numpy as np

# XOR dataset
X = /*give input values */
y = /* give true output values */

# MLP model
mlp = MLPClassifier(hidden_layer_sizes=(2,),
                activation='logistic',
                solver='sgd',
                learning_rate_init=0.5,
                max_iter=5000,
                random_state=1)

mlp.fit(X, y)

pred = mlp.predict(X)

print("Predicted Output:", pred)
print("Actual Output   :", y)

plt.plot(mlp.loss_curve_)
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Learning Curve of MLP for XOR Problem")
plt.show()
```

Sample 4 – MLP Decision Boundary Plot

```
//* Complete mlp.fit() before this code *//
x_min, x_max = -0.5, 1.5
y_min, y_max = -0.5, 1.5
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300),
                     np.linspace(y_min, y_max, 300))

# Predict over grid
Z = mlp.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot decision boundary
plt.contourf(xx, yy, Z, alpha=0.3)
plt.scatter(X[:, 0], X[:, 1])
plt.xlabel("x1")
plt.ylabel("x2")
plt.title("Decision Boundary of MLP for XOR Problem")
plt.show()
```

Sample 5: IRIS dataset MLP

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
# Load dataset
# load_iris()
# X = iris input data
# y = iris target output
# Train-test split
mlp = MLPClassifier(hidden_layer_sizes=(10,),
                    activation='relu',
                    max_iter=1000,
                    random_state=1)
mlp.fit(X_train, y_train)

# Predict
y_pred = mlp.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
plt.plot(mlp.loss_curve_)
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Learning Curve of MLP")
plt.show()
```

Exercises

1. Plot Decision boundary for Sample 5
2. Change the activation function of sample 4 and 5 to be all the below and draw the decision boundary and loss curve
    - Tanh
    - Sigmoid
    - ReLU (only for sample 4)

3. Design and implement an MLP to classify tumors as benign or malignant using the Breast Cancer Wisconsin dataset. Evaluate accuracy and plot the learning curve. Also plot the decision boundary

4. Implement an MLP to classify handwritten digits using the MNIST dataset and analyze the loss curve, decision boundary

5. Design an MLP to predict house prices using a given dataset. Plot training loss and comment on convergence.