Session 11: Introduction to Machine Learning

About Machine Learning (History and Definition)

Machine Learning (ML) is a subset of artificial intelligence (AI) that focuses on creating systems that can learn and improve from experience without explicitly being programmed. The term was coined by Arthur Samuel in 1959. Early developments in ML were primarily theoretical, but the advent of big data and increased computational power in recent decades has propelled it into practical applications.

Definition: Machine Learning is the study of algorithms and statistical models that enable computers to perform tasks by learning patterns from data.

Types of ML

1. Supervised Machine Learning

Supervised ML uses labeled data to train models. The algorithm learns the mapping between inputs and outputs.

Examples:

- Predicting house prices based on features (regression)
- Classifying emails as spam or not spam (classification)

Code Example (Python):

```
from sklearn.linear_model import LinearRegression
import numpy as np

# Sample data
X = np.array([[1], [2], [3], [4]])  # Feature
y = np.array([2, 4, 6, 8])  # Target

# Train the model
model = LinearRegression()
model.fit(X, y)

# Make a prediction
```

```
print(model.predict([[5]]))  # Output: [10]
```

2. Unsupervised Machine Learning

Unsupervised ML works with unlabeled data to find patterns and structure.

Examples:

- Clustering customers by purchasing behavior
- Dimensionality reduction

Code Example (Python):

```
from sklearn.cluster import KMeans

# Sample data
X = [[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]]

# Train the model
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
print(kmeans.labels_)  # Output: Cluster labels
```

3. Semi-Supervised Machine Learning

Combines a small amount of labeled data with a large amount of unlabeled data. Useful when labeling data is expensive or time-consuming.

4. Reinforcement Learning

Focuses on decision-making tasks where an agent learns by interacting with an environment to maximize a reward.

Examples:

- Game playing (e.g., AlphaGo)
- Robotics

Code Example (Python) (Pseudo-code for simplicity):

```
# Pseudo-code for Q-learning
for each episode:
    state = env.reset()
    while not done:
        action = choose_action(state)
        next_state, reward, done = env.step(action)
        update_Q_table(state, action, reward, next_state)
        state = next_state
```

Batch/Offline Machine Learning

In batch learning, the model is trained on the entire dataset at once. It does not update incrementally.

Disadvantages of Batch Learning

- Computationally expensive for large datasets
- Inefficient for scenarios where data is continuously updated

Online Machine Learning

Online ML trains the model incrementally, using one data instance at a time or a mini-batch.

Importance

- Suitable for dynamic environments
- Efficient for large, continuous data streams

When to Use and How to Use

- Use when data is continuously generated
- Train incrementally with `partial_fit()` in libraries like scikit-learn

Learning Rate

Determines how quickly the model adapts. A smaller learning rate ensures stability, while a larger one speeds up learning but risks overshooting.

Out-of-Core Learning

Used to handle datasets that are too large to fit in memory.

Code Example (Python):

```
from sklearn.linear_model import SGDRegressor

# Sample data stream
X = [[1], [2], [3]]
y = [2, 4, 6]

model = SGDRegressor()
for i in range(len(X)):
    model.partial_fit([X[i]], [y[i]])

print(model.coef_)  # Output: Trained coefficients
```

Disadvantages

- May require careful tuning of hyperparameters like learning rate
- Can overfit or underfit easily

Batch vs Online Learning

Aspect	Batch Learning	Online Learning
Data Processing	Entire dataset at once	Incremental updates
Efficiency	Less efficient for large data	Efficient for streaming data
Adaptability	Static	Dynamic

Instance-Based Learning

Learns by memorizing training instances and comparing new data to them.

Examples: k-Nearest Neighbors (k-NN)

Code Example (Python):

```
from sklearn.neighbors import KNeighborsClassifier

# Sample data
X = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]

# Train and predict
model = KNeighborsClassifier(n_neighbors=2)
model.fit(X, y)
print(model.predict([[1.5]]))  # Output: [0]
```

Model-Based Learning

Learns a model to generalize the training data.

Examples: Linear Regression, Neural Networks

Instance vs Model-Based Learning

Aspect	Instance-Based Learning	Model-Based Learning
Generalization	Poor	Good
Computation at Test	High	Low

Challenges in ML

1. Data Collection

Quality and quantity of data directly affect model performance.

2. Insufficient/Labelled Data

A lack of labeled data hinders supervised learning.

3. Non-Representative Data

Data that does not represent the real-world distribution can lead to biased models.

4. Poor Quality Data

Noisy or incomplete data impacts the model's ability to learn.

5. Irrelevant Features

Feature engineering is critical for meaningful predictions.

6. Overfitting and Underfitting

- Overfitting: Model performs well on training data but poorly on new data.
- Underfitting: Model fails to capture the underlying pattern.

7. Offline Learning

Cannot adapt to new data in real-time.

8. Cost

Training large models can be computationally expensive.

Machine Learning Development Life-Cycle

1. **Problem Definition**: Define the objective and success criteria.
2. **Data Collection**: Gather and preprocess data.
3. **Data Exploration**: Understand data distribution and relationships.
4. **Model Selection**: Choose the appropriate algorithm.
5. **Training**: Train the model on data.
6. **Evaluation**: Test the model's performance.
7. **Deployment**: Integrate the model into the application.
8. **Monitoring**: Continuously track and improve performance.