Unit 6: Python and Machine Learning Foundations - Theory Answers

Q1: Explain how arrays in NumPy are different from Python lists and why this is beneficial in ML

NumPy Arrays vs. Python Lists:

- **Data Type Uniformity:** NumPy arrays contain elements of the same data type, ensuring faster computations. Python lists can contain mixed data types.
- **Memory Efficiency:** Arrays are stored in contiguous memory blocks, which boosts performance. Lists are less efficient due to their dynamic and flexible nature.
- **Speed:** NumPy uses optimized C code behind the scenes, offering faster computation than standard Python loops.
- **Vectorization:** NumPy supports operations over entire arrays without explicit loops, which is useful for mathematical operations in machine learning.

Benefits in Machine Learning:

- Handling large datasets and matrices efficiently.
- Performing linear algebra, which is essential for ML algorithms.
- Faster model training due to efficient numerical computations.

Example:

```
import numpy as np

x = np.array([1, 2, 3])

y = x * 2  # Multiplies each element
```

(Refer to Python/Numpy class notes.)

Q2: Explain the role of NumPy in machine learning applications with an example.

Role of NumPy in ML:

- Provides efficient storage and operations for large datasets.
- Supports array-based computation, matrix multiplication, and statistical operations.
- Foundation for libraries like Pandas, Scikit-learn, TensorFlow.

Use Case Example: Training a machine learning model involves computing the dot product of matrices (inputs × weights). NumPy makes it efficient:

```
import numpy as np

X = np.array([[1, 2], [3, 4]])

W = np.array([[0.5], [1.5]])

output = np.dot(X, W)
```

This matrix multiplication is fundamental in neural networks and regression models.

(Refer to NumPy slide exercises.)

Q3: Explain how to install Python and set up the PATH variable in a Windows environment.

Steps to Install Python and Set PATH on Windows:

1. **Download:** Visit python.org and download the latest version.
2. **Install:** Run the installer and select "Add Python to PATH" before proceeding.
3. **Verify Installation:**
    - Open Command Prompt and type `python --version`.
    - It should display the installed version.
4. **Manually Setting PATH (if unchecked during install):**
    - Right-click on **This PC → Properties → Advanced System Settings**.
    - Go to **Environment Variables**.
    - Under **System Variables**, find `Path` → Edit → Add the Python installation path (e.g., `C:\Python39\` and `C:\Python39\Scripts\`).

Python is now ready for development.

(Refer to installation guide in lab manual.)

Q4: Explain how Python is used to implement basic AI applications.

Python is the preferred language for AI due to its simplicity and powerful libraries. It supports rapid development, testing, and deployment of AI systems.

Common Libraries in AI:

- **NumPy & Pandas:** Data handling and preprocessing.
- **Scikit-learn:** Machine learning models.
- **TensorFlow / PyTorch:** Deep learning.
- **NLTK / spaCy:** Natural Language Processing.

Example Application: Spam email classification using Scikit-learn:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train, y_train)
```

```
pred = model.predict(X_test)
```

(Refer to Python-based ML notebook.)

Q5: What is the significance of the Bellman equation in Adaptive Dynamic Programming

The **Bellman equation** is central to **Adaptive Dynamic Programming (ADP)** and reinforcement learning.

Significance:

- Describes the relationship between the **value of a state** and the **rewards** of subsequent states.
- Provides a recursive formula to calculate optimal utilities:

  $U(s) = R(s) + \gamma \Sigma P(s'|s, a) U(s')$
- Helps update utilities based on current state, reward, and expected future utility.
- Enables value iteration and policy improvement in ADP.

Application: Used by agents to learn optimal policies in stochastic environments.

(Refer to Unit 5, slides 24–25.)

Q6: Explain how passive reinforcement learning differs from active reinforcement learning with suitable examples

Passive Reinforcement Learning:

- The agent **follows a fixed policy**.
- Learns utility of states without choosing actions.
- No exploration, only evaluation.
- **Example:** A robot observes others navigating a maze to learn state values.

Active Reinforcement Learning:

- The agent **chooses its own actions** to learn the optimal policy.
- Involves exploration and exploitation.
- **Example:** A robot learns table tennis by playing games and adjusting strategy.

Key Difference:

- Passive RL focuses on evaluating a given policy.
- Active RL focuses on discovering the best policy.

(Refer to Unit 5, slides 18–21.)