DAY 21 **[18 - JULY - 2025]**

Model Evaluation and Validation

Overfitting, Underfitting

Introduction

In machine learning, achieving a model that generalizes well to unseen data is crucial. Overfitting and underfitting are common challenges that affect model performance. The bias-variance tradeoff provides a framework to understand these issues, while regularization techniques help mitigate them. This document explores these concepts, their implications, and practical approaches to address them using Python.

1. Overfitting

Definition

Overfitting occurs when a model learns the training data too well, including noise and outliers, resulting in poor performance on unseen (test) data. The model becomes overly complex, capturing patterns that do not generalize.

Characteristics

- **Training Error**: Very low.
- **Test Error**: High.
- **Symptoms**: Excessive sensitivity to small variations in the training data.

Example

A decision tree with unlimited depth may perfectly fit the training data but fail to generalize.

2. Underfitting

Definition

Underfitting occurs when a model is too simple to capture the underlying patterns in the data, leading to poor performance on both training and test data.

Characteristics

- **Training Error**: High.
- **Test Error**: High.
- **Symptoms**: Model fails to learn even the basic trends in the data.

Example

A linear regression model applied to a highly non-linear dataset may underfit.

3. Bias-Variance Tradeoff

Definition

The bias-variance tradeoff is a fundamental concept that explains the balance between a model's complexity and its predictive power. The total expected error of a model can be decomposed into three components:

- **Bias**: Error due to overly simplistic assumptions (underfitting). High bias models are too rigid.
- **Variance**: Error due to sensitivity to small fluctuations in the training data (overfitting). High variance models are too flexible.
- **Irreducible Error**: Noise inherent in the data, which cannot be reduced regardless of the model.

Illustration

- **High Bias, Low Variance**: Simple models (e.g., linear regression) have consistent but inaccurate predictions.
- **Low Bias, High Variance**: Complex models (e.g., deep neural networks) fit training data well but vary significantly with different training sets.
- **Optimal Model**: Balances bias and variance to minimize total error.

Python Example: Visualizing Bias-Variance

Below is an example that demonstrates overfitting and underfitting using polynomial regression.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline

# Generate synthetic data
np.random.seed(42)
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = np.sin(X).ravel() + np.random.normal(0, 0.1, X.shape[0])

# Fit models of varying complexity
degrees = [1, 4, 15]
plt.figure(figsize=(15, 5))

for i, degree in enumerate(degrees):
    # Create polynomial regression model
    polyreg = make_pipeline(PolynomialFeatures(degree), LinearRegression())
    polyreg.fit(X, y)

    # Predict
    X_test = np.linspace(0, 5, 100)[:, np.newaxis]
    y_pred = polyreg.predict(X_test)

    # Plot
    plt.subplot(1, 3, i + 1)
    plt.scatter(X, y, color='black', s=20, label='Data')
    plt.plot(X_test, y_pred, color='blue', label=f'Degree {degree}')
```

```
    plt.title(f'Polynomial Degree {degree}')
    plt.legend()
    plt.xlabel('X')
    plt.ylabel('y')

plt.tight_layout()
plt.show()
```

Output: (A plot with three subplots showing:

- Degree 1: Underfitting (high bias, low variance).
- Degree 4: Good fit (balanced bias and variance).
- Degree 15: Overfitting (low bias, high variance).)