Day41_Overfitting_&_Underfitting_in_Machine_Learning

July 10, 2025

Day 41 Overfitting and Underfitting in Machine Learning

1 Introduction

In this notebook, we explore two important concepts in Machine Learning: **Overfitting** and **Underfitting**.

Both are related to the performance of ML models on training and test data.

Understanding and identifying these issues is essential for building accurate, generalizable models.

2 What is Underfitting?

Underfitting occurs when a model is too simple to capture the patterns in the training data.

- Low accuracy on training data
- Low accuracy on test data
- Model is not learning enough

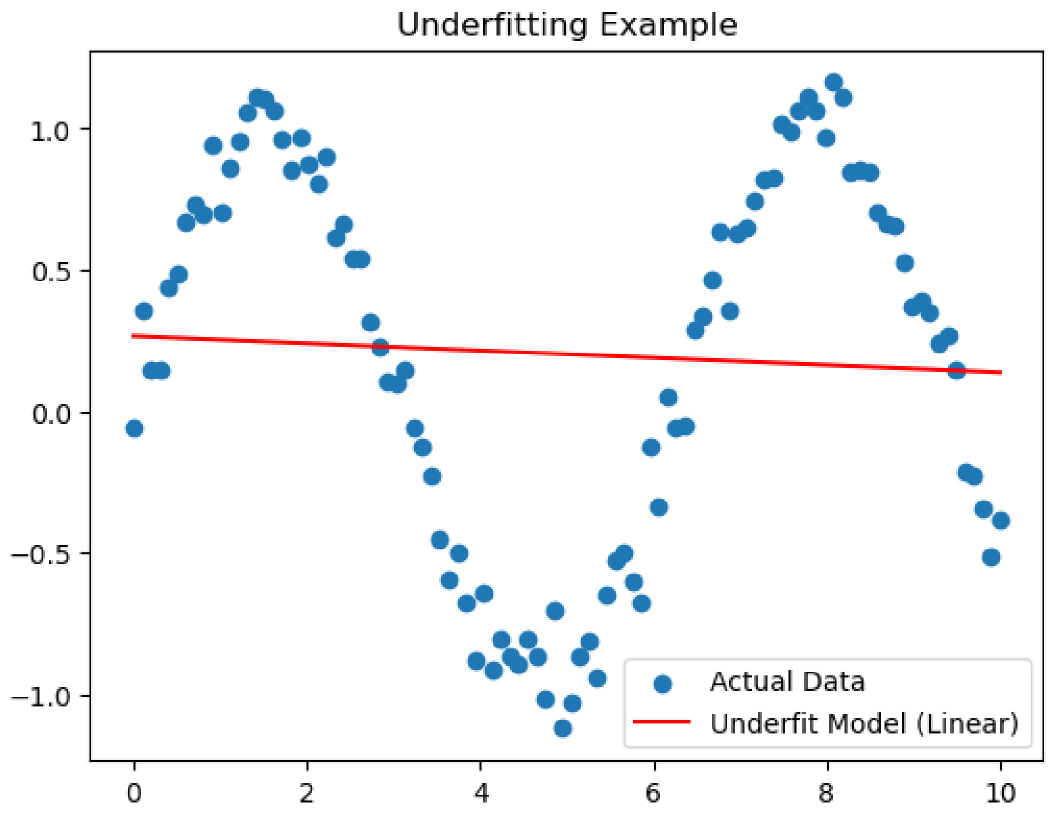
2.1 Example of Underfitting

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
```

[2]:
```
# Sample dataset (non-linear)
X = np.linspace(0, 10, 100).reshape(-1, 1)
y = np.sin(X).ravel() + np.random.normal(0, 0.1, X.shape[0])
```

[3]:
```
# Underfitting: using a very simple linear model
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)
```

[4]:
```
plt.scatter(X, y, label="Actual Data")
plt.plot(X, y_pred, color="red", label="Underfit Model (Linear)")
plt.title("Underfitting Example")
plt.legend()
plt.show()
```



[5]:
```
print("MSE:", mean_squared_error(y, y_pred))
```

```
MSE: 0.44579632914556155
```

3 What is Overfitting?

Overfitting occurs when a model is too complex and learns both the data and noise.

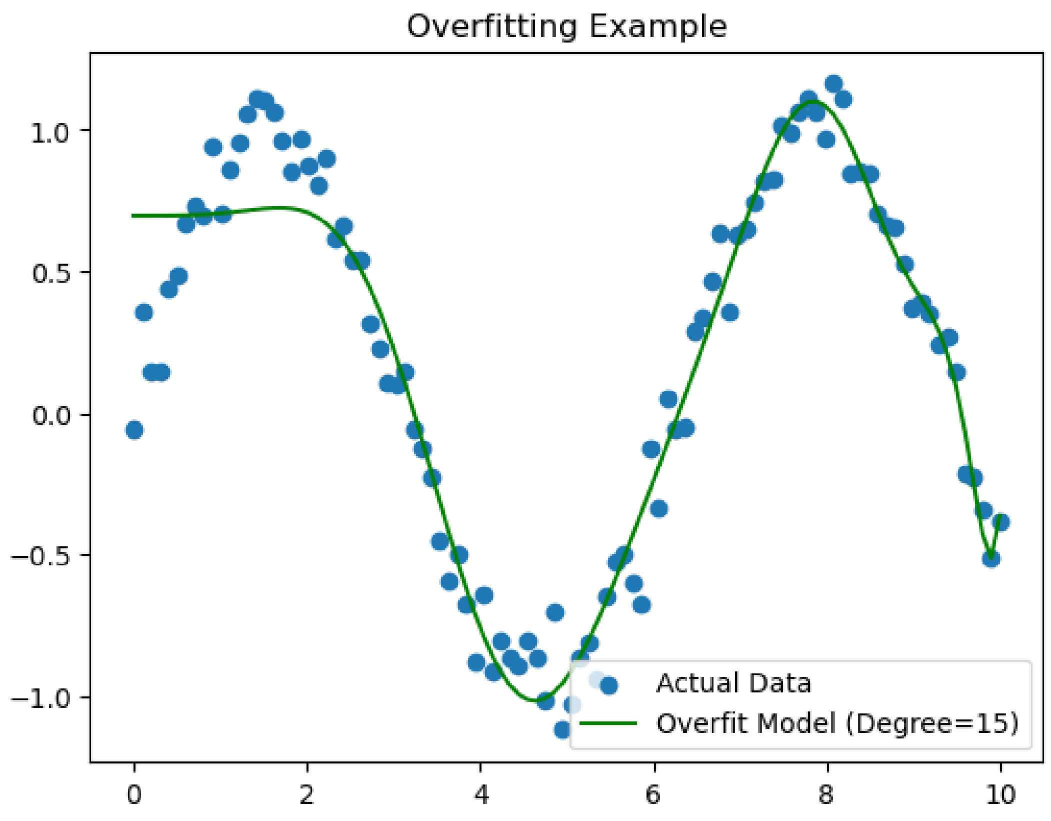
- Very high accuracy on training data
- Poor accuracy on test data
- Model memorizes instead of generalizing

3.1 Example of Overfitting with Polynomial Regression

[6]:
```
# Use Polynomial Features (high degree = overfitting)
poly = PolynomialFeatures(degree=15)
X_poly = poly.fit_transform(X)
```

[7]:
```
# Overfit model
model_overfit = LinearRegression()
model_overfit.fit(X_poly, y)
y_overfit = model_overfit.predict(X_poly)
```

[8]:
```
plt.scatter(X, y, label="Actual Data")
plt.plot(X, y_overfit, color="green", label="Overfit Model (Degree=15)")
plt.title("Overfitting Example")
plt.legend()
plt.show()
```

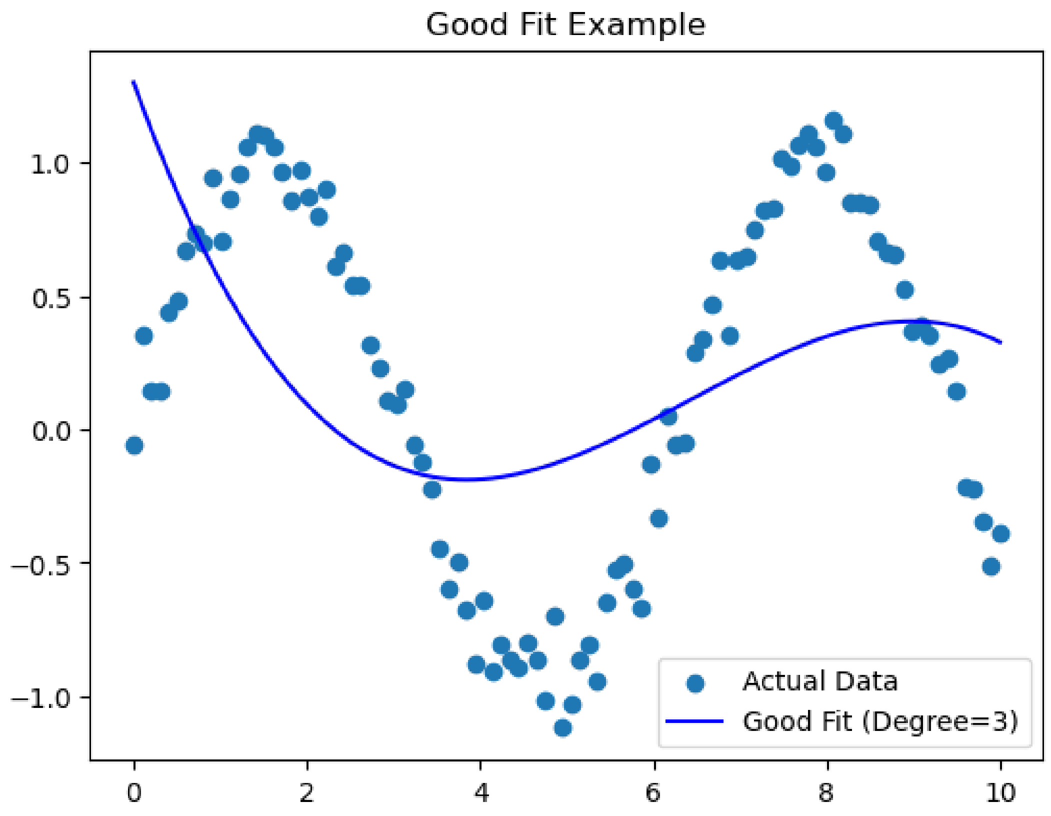


4 A Good Fit (Balanced Model)

[9]:
```
# Reasonable degree = good generalization
poly_good = PolynomialFeatures(degree=3)
X_poly_good = poly_good.fit_transform(X)
```

[10]:
```
model_good = LinearRegression()
model_good.fit(X_poly_good, y)
y_good = model_good.predict(X_poly_good)
```

[11]:
```
plt.scatter(X, y, label="Actual Data")
plt.plot(X, y_good, color="blue", label="Good Fit (Degree=3)")
plt.title("Good Fit Example")
plt.legend()
plt.show()
```



[12]:
```
print("MSE (Good Fit):", mean_squared_error(y, y_good))
```

```
MSE (Good Fit): 0.33770615037449436
```

5 How to Detect Overfitting or Underfitting?

Compare model performance:

Condition	Training Accuracy	Test Accuracy	Reason
Underfitting	Low	Low	Too simple
Overfitting	High	Low	Too complex
Good Fit	High	High	Generalizes well

6 How to Avoid Overfitting?

- Use simpler models (reduce complexity)
- Collect more training data
- Use cross-validation
- Regularization (e.g., L1, L2)
- Pruning in decision trees
- Dropout in neural networks

7 Key Takeaways

- Underfitting = model is too simple $\rightarrow$ misses important patterns.
- Overfitting = model is too complex $\rightarrow$ memorizes noise.
- Always balance the model to generalize well on unseen data.
- Use training/validation/test split and metrics like MSE or accuracy to evaluate fit.