Comprehensive Guide to Linear Regression

Osada Induwara

July 14, 2024

Table of Contents

1 Introduction

Linear regression is a fundamental algorithm in machine learning used for predicting a continuous target variable based on one or more input features. It is widely used due to its simplicity and interpretability.

2 Understanding Linear Regression

Linear regression models the relationship between a dependent variable (y) and one or more independent variables (X) by fitting a linear equation to the observed data.

3 Mathematical Foundations

3.1 Equation of a Line

The equation for a simple linear regression line is:

$$y = \beta_0 + \beta_1 X$$

Where:

- y is the dependent variable.
- X is the independent variable.
- β_0 is the intercept.
- β_1 is the slope.

3.2 Cost Function

The cost function (Mean Squared Error) measures the accuracy of the model:

$$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

3.3 Normal Equation

The normal equation provides a closed-form solution:

$$\theta = (X^T X)^{-1} X^T y$$

4 Methods for Solving Linear Regression

4.1 Normal Equation

A direct method to solve for coefficients:

$$\theta = (X^T X)^{-1} X^T y$$

4.2 Gradient Descent

An iterative optimization algorithm:

$$\theta := \theta - \alpha \frac{1}{m} \sum (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

4.3 QR Decomposition

Decomposes matrix X into Q and R:

$$X = QR$$

$$\theta = R^{-1} Q^T y$$

4.4 Singular Value Decomposition (SVD)

Factorizes matrix X into U, Σ, and V^T:

$$X = U \Sigma V^T$$

$$\theta = V \Sigma^{-1} U^T y$$

5 Practical Implementation in Python

5.1 Using NumPy

```python
import numpy as np

# Preparing data
X = np.array([2, 3, 5]).reshape(-1, 1)
y = np.array([4, 5, 7])
X_b = np.c_[np.ones((X.shape[0], 1)), X]

```

```
# Calculating coefficients
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
print(f"Intercept: {theta_best[0]}, Slope: {theta_best[1]}")
```

Listing 1: Linear Regression using NumPy

5.2 Using scikit-learn

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X, y)
print(f"Intercept: {model.intercept_}, Slope: {model.coef_[0]}")
```

Listing 2: Linear Regression using scikit-learn

5.3 Using statsmodels

```
import statsmodels.api as sm

X_b = sm.add_constant(X)
model = sm.OLS(y, X_b).fit()
print(model.summary())
```

Listing 3: Linear Regression using statsmodels

5.4 Using TensorFlow

```
import tensorflow as tf

X = tf.constant([[2.0], [3.0], [5.0]], dtype=tf.float32)
y = tf.constant([4.0, 5.0, 7.0], dtype=tf.float32)

model = tf.keras.Sequential([tf.keras.layers.Dense(1, input_shape=(1,))])
```

```
model.compile(optimizer='sgd', loss='mse')
model.fit(X, y, epochs=100)
print(f"Intercept and Slope: {model.layers[0].weights}")
```

Listing 4: Linear Regression using TensorFlow

6 Sample Projects

6.1 Simple Linear Regression with NumPy

```
import numpy as np
import matplotlib.pyplot as plt

# Data
X = np.array([1, 2, 4, 3, 5])
y = np.array([1, 3, 3, 2, 5])
X_b = np.c_[np.ones((X.shape[0], 1)), X]

# Linear Regression using Normal Equation
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)

# Prediction
X_new = np.array([[0], [6]])
X_new_b = np.c_[np.ones((2, 1)), X_new]
y_predict = X_new_b.dot(theta_best)

# Plot
plt.plot(X, y, "b.")
plt.plot(X_new, y_predict, "r-")
plt.xlabel("X")
plt.ylabel("y")
plt.show()
```

Listing 5: Simple Linear Regression with NumPy

6.2 Multiple Linear Regression with scikit-learn

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

# Data
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.dot(X, np.array([1, 2])) + 3

# Model
model = LinearRegression().fit(X, y)
print(f"Intercept: {model.intercept_}, Coefficients:
    {model.coef_}")
```

Listing 6: Multiple Linear Regression with scikit-learn

6.3 Linear Regression with TensorFlow

```python
import tensorflow as tf

# Data
X = tf.constant([[1.0], [2.0], [4.0], [3.0], [5.0]],
    dtype=tf.float32)
y = tf.constant([1.0, 3.0, 3.0, 2.0, 5.0], dtype=tf.
   float32)

# Model
model = tf.keras.Sequential([tf.keras.layers.Dense
   (1, input_shape=(1,))])
model.compile(optimizer='sgd', loss='mse')
model.fit(X, y, epochs=100)
```

Listing 7: Linear Regression with TensorFlow