Linear Regression

R. Brooks

Introduction

Linear regression is a foundational method for modelling how a continuous outcome depends on one or more input variables. The goal is to approximate the relationship between inputs and output using a linear function. Despite its simplicity, linear regression is widely used in machine learning due to its interpretability, solid baseline performance, and clear mathematical structure.

In this course, linear regression also serves as an important bridge between linear algebra and machine learning algorithms, since the entire model can be expressed and solved using matrix operations.

The Linear Regression Model

Given a dataset with n observations and p features, the linear regression model can be written as

$$\hat{y} = X\mathbf{B}$$

where:

- $X \in \mathbb{R}^{n\times(p+1)}$ is the **design matrix**,
- $\mathbf{B} \in \mathbb{R}^{(p+1)}$ is the **coefficient vector**,
- $\hat{y} \in \mathbb{R}^{n}$ contains the predicted values.

The extra column in X accounts for the intercept term, allowing the regression line (or hyperplane) to shift vertically.

The Normal Equation

Linear regression finds the coefficient vector $\mathbf{B}$ that minimizes the **sum of squared errors** between the true targets y and the predictions $\hat{y}$. This optimization problem has a closed-form solution known as the **normal equation**:

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

This equation directly computes the coefficients that best fit the data in a least-squares sense.

Meaning of Each Term

- $X^T X$ captures how the features relate to each other (feature correlation).
- $X^T y$ captures how each feature relates to the target variable.
- $(X^T X)^{-1}$ rescales these relationships to produce optimal coefficients.

Intuitively, the normal equation balances the contribution of each feature so that the overall squared prediction error is minimized.

Step-by-Step Explanation

1. **Construct the design matrix X:**
   - Each **row** corresponds to one data point.
   - Each **column** corresponds to one feature.
   - The **first column consists of ones**, allowing the model to learn an intercept term.
2. **Compute the normal equation:**
   - Compute $X^T X$.
   - Compute $X^T y$.
   - Invert $X^T X$ and multiply by $X^T y$ to obtain $\mathbf{B}$.
3. **Interpret the coefficients:**
   - The first entry in $\mathbf{B}$ is the **intercept**.
   - Each remaining entry represents the expected change in y when the corresponding feature increases by one unit (holding others constant).

Evaluation of the Model

Once the regression coefficients $\mathbf{B}$ have been computed, we assess the quality of the model using quantitative evaluation metrics.

Mean Squared Error (MSE)

The Mean Squared Error measures the average squared difference between true values and predictions:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

A lower MSE indicates that predictions are, on average, closer to the true values.

R^2 (Coefficient of Determination)

The R^2 score measures how much of the variance in the target variable is explained by the model:

$$R^2 = 1 - \frac{\sum(y - \hat{y})^2}{\sum(y - \bar{y})^2}$$

where $\bar{y}$ is the mean of the target values.

Computing the Metrics

1. **Compute predictions:**
$$\hat{y} = X\mathbf{B}$$
2. **Compute residuals:**
$$y - \hat{y}$$
3. **Compute MSE** by squaring and averaging the residuals.
4. **Compute** R^2 by comparing unexplained variance to total variance.

Interpreting the Results

- **Low MSE** indicates good predictive accuracy.
- $R^2 = 1$ indicates a perfect fit.
- $R^2 = 0$ means the model performs no better than predicting the mean of y.
- $R^2 < 0$ indicates the model performs worse than a baseline model.

Practical Considerations

If two or more features are perfectly correlated (a situation called multicollinearity) or if there are more features than data points, the matrix $\mathbf{X}^T\mathbf{X}$ may not be invertible. In practice, we often use the *Pseudo-Inverse* (e.g., `pinv` in NumPy) to handle these edge cases. Additionally, for very large datasets, computing the inverse (or pseudo-inverse) can be computationally expensive.

For these reasons, machine learning libraries often use numerical methods such as gradient descent or matrix factorization instead of directly computing the inverse.