The Mathematics of Linear Regression

Hom Bahrani

2025

Abstract

This document provides a comprehensive mathematical treatment of linear regression, focusing on the underlying theory, gradient descent optimization, and cost functions. We derive the necessary equations, analyze convergence properties, and connect the statistical and geometric interpretations of linear regression.

Contents

1 Introduction to Linear Regression

Linear regression is one of the fundamental supervised learning algorithms in statistics and machine learning. At its core, it models the relationship between a dependent variable and one or more independent variables using a linear equation.

1.1 Mathematical Formulation

Given a dataset $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ consisting of m examples, where:

- $x^{(i)} \in \mathbb{R}^n$ is the feature vector for the i-th example with n features, and
- $y^{(i)} \in \mathbb{R}$ is the corresponding target value

We define the hypothesis function as:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{1}$$

This can be rewritten as:

$$h_\theta(x) = \theta^T x \tag{2}$$

where we define:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad \text{and} \quad x = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{3}$$

Note that we've added a bias term θ_0, and correspondingly, added a feature $x_0 = 1$ to accommodate this bias term.

1.2 Key Assumptions of Linear Regression

Linear regression makes several key assumptions:

1. **Linearity**: The relationship between the features and the target is linear.
2. **Independence**: The errors (residuals) are independent of each other.
3. **Homoscedasticity**: The errors have constant variance.
4. **Normality**: The errors are normally distributed.
5. **No multicollinearity**: The features are not highly correlated with each other.

1.3 Matrix Formulation

For a dataset with m examples and n features, we can express the problem in matrix form:

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \quad \text{and} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \tag{4}$$

The hypothesis function for all examples can be written as:

$$h_\theta = X\theta \tag{5}$$

2 Cost Functions for Linear Regression

A cost function measures how well our model fits the data. In the context of linear regression, we need to quantify the difference between our predictions and the actual values.

2.1 Mean Squared Error (MSE)

The most common cost function for linear regression is the Mean Squared Error (MSE):

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{6}$$

In matrix notation:

$$J(\theta) = \frac{1}{2m} (X\theta - y)^T (X\theta - y) \tag{7}$$

We expand the matrix form:

$$\begin{aligned}(X\theta - y)^T (X\theta - y) &= (X\theta)^T (X\theta) - (X\theta)^T y - y^T (X\theta) + y^T y && (8)\\ &= \theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y && (9)\\ &= \theta^T X^T X\theta - 2\theta^T X^T y + y^T y && (10)\\ & && (11)\end{aligned}$$

This is equivalent to the sum of squared residuals:

$$\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{12}$$

The factor of $\frac{1}{2}$ is added for mathematical convenience (when we take the derivative, the 2 in the exponent and the $\frac{1}{2}$ cancel out).

2.2 Mean Absolute Error (MAE)

An alternative cost function is the Mean Absolute Error (MAE):

$$J_{MAE}(\theta) = \frac{1}{m} \sum_{i=1}^{m} |h_\theta(x^{(i)}) - y^{(i)}| \tag{13}$$

Unlike MSE, MAE is less sensitive to outliers but is not differentiable at all points, which makes optimization more challenging.

2.3 Huber Loss

Huber loss combines the best properties of MSE and MAE:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta \\ \delta(|a| - \frac{1}{2}\delta) & \text{for } |a| > \delta \end{cases} \tag{14}$$

where $a = h_\theta(x^{(i)}) - y^{(i)}$ is the residual and δ is a hyperparameter. Huber loss is quadratic for small residuals and linear for large residuals, making it robust to outliers while remaining differentiable.

2.4 Probabilistic Interpretation

Linear regression can be derived from a probabilistic perspective. If we assume that:

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)} \tag{15}$$

where $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise, then:

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \tag{16}$$

The likelihood of the entire dataset is:

$$L(\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) \tag{17}$$

Taking the negative log-likelihood and dropping constant terms, we get:

$$-\log L(\theta) \propto \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2 \tag{18}$$

This is equivalent to the MSE cost function! Thus, minimizing the MSE is equivalent to maximizing the likelihood under Gaussian noise assumptions.

3 Analytical Solution to Linear Regression

Before diving into gradient descent, it's worth noting that linear regression has a closed-form solution.

3.1 Normal Equation

For MSE, we can find the optimal parameters by setting the gradient to zero:

$$\nabla_\theta J(\theta) = \frac{1}{m} X^T (X\theta - y) = 0 \tag{19}$$

Solving for θ:

$$X^T X \theta = X^T y \tag{20}$$

If $X^T X$ is invertible, the solution is:

$$\theta = (X^T X)^{-1} X^T y \tag{21}$$

This is known as the Normal Equation. It provides an exact solution without requiring iteration, but can be computationally expensive for large datasets due to the matrix inversion.

3.2 Limitations of the Analytical Solution

The analytical solution has several limitations:

- Computing $(X^T X)^{-1}$ has time complexity $O(n^3)$, where n is the number of features.
- For large n, the inversion becomes computationally prohibitive.
- If $X^T X$ is singular (non-invertible), we face the problem of multicollinearity.

4 Gradient Descent Optimization

Gradient descent is an iterative optimization algorithm for finding the minimum of a differentiable function. It's particularly useful for large datasets where the analytical solution becomes computationally expensive.

4.1 The Gradient Descent Algorithm

The core idea of gradient descent is to iteratively update the parameters in the direction of steepest descent (negative gradient):

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \tag{22}$$

where α is the learning rate.

4.2 Derivation of the Gradient for MSE

Let's derive the gradient of the MSE cost function:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{23}$$

The partial derivative with respect to θ_j is:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{24}$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\frac{\partial}{\partial \theta_j}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{25}$$

$$= \frac{1}{2m}\sum_{i=1}^{m}2(h_\theta(x^{(i)}) - y^{(i)})\frac{\partial}{\partial \theta_j}(h_\theta(x^{(i)}) - y^{(i)}) \tag{26}$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\frac{\partial}{\partial \theta_j}(h_\theta(x^{(i)})) \tag{27}$$

$$\tag{28}$$

Since $h_\theta(x^{(i)}) = \theta^T x^{(i)} = \sum_{k=0}^{n}\theta_k x_k^{(i)}$, we have:

$$\frac{\partial h_\theta(x^{(i)})}{\partial \theta_j} = x_j^{(i)} \tag{29}$$

Therefore:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \tag{30}$$

In vector form, the gradient is:

$$\nabla_\theta J(\theta) = \frac{1}{m}X^T(X\theta - y) \tag{31}$$

4.3 Gradient Descent Variants

There are several variants of gradient descent:

4.3.1 Batch Gradient Descent

The standard gradient descent algorithm updates parameters using the entire dataset:

$$\theta := \theta - \alpha\nabla_\theta J(\theta) \tag{32}$$

This can be computationally expensive for large datasets.

4.3.2 Stochastic Gradient Descent (SGD)

SGD updates parameters using a single randomly selected example:

$$\theta := \theta - \alpha\nabla_\theta J_i(\theta) \tag{33}$$

where $J_i(\theta) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$ is the cost for a single example.
This is much faster but can lead to noisy updates.

4.3.3 Mini-Batch Gradient Descent

Mini-batch gradient descent combines the best of both worlds by updating parameters using a small random subset (mini-batch) of the data:

$$\theta := \theta - \alpha \nabla_\theta J_B(\theta) \tag{34}$$

where $J_B(\theta)$ is the cost for a mini-batch B.

4.4 Convergence Analysis

The convergence of gradient descent depends on several factors:

4.4.1 Learning Rate

The learning rate α determines the step size in each iteration:

- If α is too small, convergence will be slow.
- If α is too large, the algorithm may oscillate or diverge.

For convex functions like MSE, there exists a theoretical upper bound on the learning rate to ensure convergence. If λ_{max} is the largest eigenvalue of the Hessian matrix $\nabla^2 J(\theta)$, then convergence is guaranteed if:

$$0 < \alpha < \frac{2}{\lambda_{max}} \tag{35}$$

4.4.2 Convergence Rate

For a convex differentiable function with Lipschitz continuous gradients, batch gradient descent converges at a rate of $O(1/k)$, where k is the number of iterations. That is:

$$J(\theta^{(k)}) - J(\theta^*) \leq \frac{C}{k} \tag{36}$$

where θ^* is the optimal parameter and C is a constant.

5 Feature Normalization

Feature normalization is crucial for the efficient performance of gradient descent, especially when features have different scales.

5.1 Standardization

Standardization transforms features to have zero mean and unit variance:

$$x_{norm} = \frac{x - \mu}{\sigma} \tag{37}$$

where μ is the mean and σ is the standard deviation of the feature.

5.2 Min-Max Scaling

Min-max scaling transforms features to a specific range, typically [0, 1]:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{38}$$

5.3 Mathematical Justification

Consider a simple two-feature problem with MSE cost function. The gradient descent update for the first parameter is:

$$\theta_1 := \theta_1 - \alpha \frac{\partial J(\theta)}{\partial \theta_1} \tag{39}$$

If feature x_1 has a much larger scale than x_2, the partial derivative $\frac{\partial J(\theta)}{\partial \theta_1}$ will be larger, causing the gradient descent to oscillate along the θ_1 axis, requiring a smaller learning rate and more iterations to converge.

Geometrically, this creates an elongated contour plot of the cost function. Normalization transforms these contours to be more circular, allowing gradient descent to converge more efficiently.

6 Regularization in Linear Regression

Regularization helps prevent overfitting by adding a penalty term to the cost function that discourages large parameter values.

6.1 Ridge Regression (L2 Regularization)

Ridge regression adds a sum of squared parameter values to the cost function:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \tag{40}$$

The regularization term excludes the bias parameter θ_0. In matrix form:

$$J(\theta) = \frac{1}{2m}(X\theta - y)^T(X\theta - y) + \frac{\lambda}{2m} \theta_{-0}^T \theta_{-0} \tag{41}$$

where θ_{-0} denotes θ without the bias term.

The gradient becomes:

$$\nabla_\theta J(\theta) = \frac{1}{m} X^T(X\theta - y) + \frac{\lambda}{m} \theta' \tag{42}$$

where θ' is θ with the bias term set to zero.

The analytical solution is:

$$\theta = (X^T X + \lambda I')^{-1} X^T y \tag{43}$$

where I' is the identity matrix with the first diagonal element set to zero (to exclude regularization on the bias term).

6.2 Lasso Regression (L1 Regularization)

Lasso regression adds a sum of absolute parameter values:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{m}\sum_{j=1}^{n}|\theta_j| \tag{44}$$

The L1 penalty term encourages sparsity (many parameters becoming exactly zero), effectively performing feature selection.

6.3 Elastic Net

Elastic Net combines L1 and L2 penalties:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{m}\left[\alpha\sum_{j=1}^{n}|\theta_j| + \frac{1-\alpha}{2}\sum_{j=1}^{n}\theta_j^2\right] \tag{45}$$

where $\alpha \in [0, 1]$ controls the balance between L1 and L2 regularization.

7 Statistical Properties of Linear Regression

Linear regression has several important statistical properties when certain assumptions are met.

7.1 Unbiasedness of OLS Estimator

The Ordinary Least Squares (OLS) estimator $\hat{\theta} = (X^TX)^{-1}X^Ty$ is unbiased:

$$E[\hat{\theta}] = \theta \tag{46}$$

Assuming $y = X\theta + \epsilon$ and $E[\epsilon] = 0$, we have:

$$\begin{aligned}
E[\hat{\theta}] &= E[(X^TX)^{-1}X^Ty] && (47)\\
&= E[(X^TX)^{-1}X^T(X\theta + \epsilon)] && (48)\\
&= E[(X^TX)^{-1}X^TX\theta + (X^TX)^{-1}X^T\epsilon] && (49)\\
&= E[\theta + (X^TX)^{-1}X^T\epsilon] && (50)\\
&= \theta + (X^TX)^{-1}X^TE[\epsilon] && (51)\\
&= \theta + (X^TX)^{-1}X^T \cdot 0 && (52)\\
&= \theta && (53)
\end{aligned}$$

7.2 Variance of OLS Estimator

The variance-covariance matrix of the OLS estimator is:

$$Var(\hat{\theta}) = \sigma^2 (X^T X)^{-1} \tag{54}$$

where σ^2 is the variance of the error term ϵ.

7.3 Gauss-Markov Theorem

Under the classical linear model assumptions, the OLS estimator is the Best Linear Unbiased Estimator (BLUE), meaning it has the smallest variance among all unbiased linear estimators.

7.4 Confidence Intervals for Parameters

Assuming errors are normally distributed, we can construct confidence intervals for the parameters. For the j-th parameter:

$$\hat{\theta}_j \pm t_{n-p,1-\alpha/2} \cdot \hat{\sigma}\sqrt{(X^T X)^{-1}_{jj}} \tag{55}$$

where:

- $t_{n-p,1-\alpha/2}$ is the critical value from the t-distribution with $n-p$ degrees of freedom
- $\hat{\sigma}$ is the estimated standard deviation of the error term
- $(X^T X)^{-1}_{jj}$ is the j-th diagonal element of $(X^T X)^{-1}$

7.5 Goodness-of-Fit Measures

7.5.1 Coefficient of Determination (R^2)

R^2 measures the proportion of variance in the dependent variable that is predictable from the independent variables:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_i (y^{(i)} - \hat{y}^{(i)})^2}{\sum_i (y^{(i)} - \bar{y})^2} \tag{56}$$

where SS_{res} is the sum of squared residuals and SS_{tot} is the total sum of squares.

7.5.2 Adjusted R^2

Adjusted R^2 accounts for the number of predictors:

$$R^2_{adj} = 1 - \frac{(1 - R^2)(n-1)}{n-p-1} \tag{57}$$

where n is the number of samples and p is the number of predictors (excluding the constant term).

8 Computational Implementation of Linear Regression

Let's outline the algorithm for implementing linear regression using gradient descent:

Algorithm 1 Linear Regression with Gradient Descent

```
1:  Input: Training data $(X, y)$, learning rate $\alpha$, iterations $T$
2:  Output: Model parameters $\theta$
3:  Normalize features: $X_{norm} = \frac{X-\mu}{\sigma}$
4:  Initialize $\theta = \vec{0}$
5:  for $t = 1$ to $T$ do
6:      $\hat{y} = X_{norm}\theta$
7:      $\nabla J = \frac{1}{m} X_{norm}^T (\hat{y} - y)$
8:      $\theta = \theta - \alpha \nabla J$
9:  end for
10: Convert $\theta$ back to original scale
11: return $\theta$
```

8.1 Converting Parameters Back to Original Scale

After training with normalized features, we need to convert the parameters back to the original scale:

$$h_\theta(x) = \theta_0 + \theta_1 \frac{x_1 - \mu_1}{\sigma_1} + \theta_2 \frac{x_2 - \mu_2}{\sigma_2} + \ldots + \theta_n \frac{x_n - \mu_n}{\sigma_n} \tag{58}$$

$$= \theta_0 - \sum_{j=1}^{n} \theta_j \frac{\mu_j}{\sigma_j} + \sum_{j=1}^{n} \frac{\theta_j}{\sigma_j} x_j \tag{59}$$

The original parameters are:

$$\theta_0' = \theta_0 - \sum_{j=1}^{n} \theta_j \frac{\mu_j}{\sigma_j} \tag{60}$$

$$\theta_j' = \frac{\theta_j}{\sigma_j} \quad \text{for } j = 1, 2, \ldots, n \tag{61}$$

8.2 Numerical Stability Considerations

When implementing linear regression, several numerical issues can arise:

- **Ill-conditioned matrices**: If $X^T X$ is nearly singular, numerical instabilities can occur in the analytical solution. Using QR decomposition or SVD can mitigate this.

- **Overflow/underflow**: Extreme values can cause numerical issues. Proper normalization helps prevent this.
- **Convergence monitoring**: Gradient descent should be monitored for convergence. Common criteria include:
  - Small change in cost: $|J(\theta^{(t+1)}) - J(\theta^{(t)})| < \epsilon$
  - Small gradient norm: $\|\nabla J(\theta^{(t)})\| < \epsilon$
  - Maximum iterations reached

9 Conclusion and Advanced Topics

Linear regression serves as a foundation for many advanced machine learning techniques. While we've covered the core mathematics, several advanced topics extend from here:

- **Polynomial Regression**: Extends linear regression to model nonlinear relationships by including polynomial terms.
- **Generalized Linear Models**: Extends linear regression to non-Gaussian error distributions.
- **Multiple Output Regression**: Predicts multiple target variables simultaneously.
- **Non-parametric Regression**: Makes fewer assumptions about the underlying data distribution.
- **Bayesian Linear Regression**: Places prior distributions on the parameters and computes posterior distributions.

Understanding the mathematical foundations of linear regression provides insight into more complex algorithms and helps in making informed choices about model selection, optimization techniques, and regularization strategies.

References

[1] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

[2] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer.

[3] Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.