Linear Methods for Regression

Ai for Quant Research
AiQR Academy

December 15, 2025

Contents

1 Linear Regression

Linear regression is a fundamental algorithm used to predict a continuous target variable based on one or more input features. The aim is to model the relationship between the dependent variable y and the independent variables $X = [x_1, x_2, \ldots, x_n]$ using a linear function.

2 The Least Mean Squares Algorithm

The Least Mean Squares (LMS) algorithm is an iterative optimization method that minimizes the cost function for linear regression by adjusting the model parameters. It is based on gradient descent, which updates the parameters to reduce the error between predicted and actual values.

2.1 Cost Function

The cost function in linear regression is defined as the Mean Squared Error (MSE):

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

Where:

- m is the number of training examples,
- $h_\theta(x^{(i)})$ is the predicted value for the i-th example, given by the hypothesis:

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)}$$

- $y^{(i)}$ is the actual observed value.

The objective of the LMS algorithm is to minimize $J(\theta)$, leading to optimal values of θ.

2.2 Gradient Descent

Gradient descent is the method used by LMS to update the parameters by moving in the opposite direction of the gradient (the slope) of the cost function with respect to the parameters.

The update rule for gradient descent is given by:

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

Where:

- α is the learning rate, controlling the step size.
- $\frac{\partial}{\partial \theta_j} J(\theta)$ is the gradient of the cost function with respect to θ_j.

2.2.1 Derivation of the Gradient

We compute the partial derivative of $J(\theta)$ with respect to θ_j:

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

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

Since $h_\theta(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)}$, the derivative of $h_\theta(x^{(i)})$ with respect to θ_j is $x_j^{(i)}$:

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

Thus, the gradient simplifies to:

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

This equation shows how the parameter θ_j is adjusted based on the error between the predicted and actual output, weighted by the feature $x_j^{(i)}$.

2.3 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is an alternative to batch gradient descent that updates the parameters after each training example rather than after the entire dataset.

2.3.1 SGD Update Rule

The update rule for SGD is:

$$\theta_j := \theta_j - \alpha \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Where $(x^{(i)}, y^{(i)})$ represents a single training example. This update is performed for each training example, making SGD faster than batch gradient descent for large datasets.

2.3.2 Advantages of SGD

- **Speed:** Parameters are updated more frequently, making the learning process faster for large datasets.
- **Real-time learning:** SGD is useful for online learning, where the model can be updated continuously as new data arrives.

2.3.3 Disadvantages of SGD

- **Noisy updates:** Since each update is based on a single example, the updates can be noisy, leading to fluctuations in the cost function.
- **Non-smooth convergence:** The cost function may not decrease smoothly as in batch gradient descent.

2.3.4 Mini-Batch Gradient Descent

A compromise between batch gradient descent and stochastic gradient descent is mini-batch gradient descent, where updates are made after processing small batches of data. This reduces the noise in the updates while allowing for more frequent parameter updates than batch gradient descent.

3 The Normal Equations

The Normal Equations provide an analytical solution to linear regression by minimizing the cost function without iterative methods like gradient descent. It is derived from the Least Squares criterion.

Given the cost function $J(\theta)$, the normal equation is obtained by finding the values of θ that minimize this cost function.

3.1 Cost Function in Matrix Form

We define the cost function as the sum of squared residuals (errors between predicted and actual values):

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

Where:

- $\mathbf{X}$ is the matrix of input features (size $m \times n$),
- θ is the vector of parameters (size $n \times 1$),
- $\mathbf{y}$ is the vector of actual outputs (size $m \times 1$),
- m is the number of training examples.

The goal is to minimize $J(\theta)$ by solving for θ.

3.2 Deriving the Normal Equation

To minimize $J(\theta)$, we compute its gradient with respect to θ:

$$\begin{aligned}
\frac{\partial J(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j}\left(\frac{1}{2m}(X\theta - \mathbf{y})^T(X\theta - \mathbf{y})\right) \\
&= \frac{1}{2m}\frac{\partial}{\partial \theta_j}\left((X\theta)^T X\theta - (X\theta)^T\mathbf{y} - \mathbf{y}^T(X\theta) + \mathbf{y}^T\mathbf{y}\right) \\
&= \frac{1}{2m}\frac{\partial}{\partial \theta_j}\left((X\theta)^T X\theta - \theta^T X^T\mathbf{y} - \mathbf{y}^T(X\theta) + \mathbf{y}^T\mathbf{y}\right) \\
&= \frac{1}{2m}\frac{\partial}{\partial \theta_j}\left(\theta^T(X^T X)\theta - \mathbf{y}^T(X\theta) - \mathbf{y}^T(X\theta) + \mathbf{y}^T\mathbf{y}\right) \\
&= \frac{1}{2m}\frac{\partial}{\partial \theta_j}\left(\theta^T(X^T X)\theta - \mathbf{y}^T X\theta - \mathbf{y}^T X\theta + \mathbf{y}^T\mathbf{y}\right) \\
&= \frac{1}{2m}\frac{\partial}{\partial \theta_j}\left(\theta^T(X^T X)\theta - 2(X^T\mathbf{y})^T\theta + \mathbf{y}^T\mathbf{y}\right) \\
&= \frac{1}{2m}\left(2X^T X\theta - 2X^T\mathbf{y}\right) \\
&= \frac{1}{m}\left(X^T X\theta - X^T\mathbf{y}\right)
\end{aligned}$$

Setting the gradient equal to zero to minimize the cost function:

$$\frac{\partial J(\theta)}{\partial \theta_j} = 0 \Rightarrow \mathbf{X}^T\mathbf{X}\theta = \mathbf{X}^T\mathbf{y}$$

Solving for θ, we get the Normal Equation:

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

This is the analytical solution to linear regression that minimizes the cost function $J(\theta)$.

4 Probabilistic Interpretation

Linear regression can also be viewed through a probabilistic lens. Assume the relationship between y and x is given by:

$$y = \theta^T x + \epsilon$$

Where $\epsilon \sim N(0, \sigma^2)$ is a normally distributed error term.

This assumption leads to the likelihood function for the data:

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

For n independent observations, the likelihood function becomes:

$$\mathcal{L}(\theta; X, y) = \prod_{i=1}^{n} p(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}\right)$$

4.1 Log-Likelihood Function

Taking the logarithm of the likelihood function gives us the log-likelihood:

$$\log \mathcal{L}(\theta; X, y) = \sum_{i=1}^{n} \left(-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}\right)$$

Simplifying:

$$\log \mathcal{L}(\theta; X, y) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y^{(i)}-\theta^T x^{(i)})^2$$

4.2 Maximizing the Log-Likelihood

To estimate θ, we maximize the log-likelihood with respect to θ. Since $-\frac{n}{2}\log(2\pi\sigma^2)$ is independent of θ, we focus on maximizing:

$$\log \mathcal{L}(\theta; X, y) \propto -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y^{(i)}-\theta^T x^{(i)})^2$$

Maximizing this is equivalent to minimizing the ordinary least squares cost function:

$$\sum_{i=1}^{n}(y^{(i)}-\theta^T x^{(i)})^2$$

Thus, linear regression can be interpreted as performing maximum likelihood estimation under the assumption of normally distributed errors.

5 Assumptions in Linear Regression

For the linear regression model to produce reliable estimates and valid inferences, several key assumptions must hold. These assumptions are:

- **Linearity:** The relationship between the independent variables $\mathbf{X}$ and the dependent variable y is linear. This means that the model can be written as:

  $$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

  where ϵ represents the error term.

- **Independence:** The residuals (errors) are independent of each other. There is no correlation between consecutive errors, which is crucial for time series or cross-sectional data.
- **Homoscedasticity:** The variance of the errors is constant across all levels of the independent variables. Mathematically, this implies:

  $$\text{Var}(\epsilon_i) = \sigma^2 \quad \text{for all } i$$

  This is also known as constant variance.
- **Normality of errors:** The error terms ϵ_i are normally distributed, especially for inference purposes (e.g., confidence intervals and hypothesis tests). Formally, we assume:

  $$\epsilon_i \sim N(0, \sigma^2)$$
- **No multicollinearity:** There should be no perfect linear relationship among the independent variables. Multicollinearity occurs when one predictor variable in a model can be linearly predicted from the others with a high degree of accuracy, which can affect the stability of coefficient estimates.
- **Exogeneity:** The error term ϵ is uncorrelated with the independent variables, i.e.:

  $$\mathbb{E}[\epsilon|\mathbf{X}] = 0$$

  This ensures that the independent variables are not influenced by omitted variables or measurement errors.

If these assumptions are violated, the results of the linear regression model may be biased or inefficient, potentially leading to misleading conclusions.

6 Shrinkage Methods

Shrinkage methods are regression techniques used to reduce overfitting by shrinking the regression coefficients. They add a penalty term to the cost function, which regularizes the coefficients. The two most common shrinkage methods are Ridge Regression and Lasso Regression. Elastic Net combines the two methods.

6.1 Ridge Regression

Ridge regression adds an $L2$ penalty to the ordinary least squares cost function to shrink the regression coefficients:

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

Where:

- m is the number of training examples,
- $\lambda \geq 0$ is the regularization parameter that controls the strength of the penalty,
- θ_j^2 is the L2 norm penalty.

The solution to Ridge regression is given by the closed-form equation:

$$\theta = \left(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I} \right)^{-1} \mathbf{X}^T \mathbf{y}$$

Ridge regression reduces overfitting by shrinking all coefficients, but it does not perform feature selection, meaning all features remain in the model.

6.2 Lasso Regression

Lasso regression adds an $L1$ penalty to the cost function, shrinking some coefficients exactly to zero, effectively performing feature selection:

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

Where:

- $\lambda \geq 0$ is the regularization parameter,
- $|\theta_j|$ is the L1 norm penalty.

Lasso performs feature selection by shrinking less important coefficients to exactly zero, making it useful when only a subset of features is relevant to the model.

6.3 Elastic Net

Elastic Net combines both Lasso and Ridge penalties, taking the best of both worlds:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - \theta^T x^{(i)} \right)^2 + \lambda_1 \sum_{j=1}^{n} |\theta_j| + \lambda_2 \sum_{j=1}^{n} \theta_j^2$$

Where:

- λ_1 controls the Lasso penalty,
- λ_2 controls the Ridge penalty.

Elastic Net is useful when there are correlated features and when both feature selection and coefficient shrinkage are needed. It combines the advantages of both Ridge and Lasso.

7 Remainder

7.1 Basic Matrix Derivatives

- Derivative of a linear form:

  $$\frac{\partial}{\partial \theta} (\mathbf{a}^T \theta) = \mathbf{a}$$

  Where $\mathbf{a}$ is a vector and θ is the vector of parameters.

- Derivative of a quadratic form:

  $$\frac{\partial}{\partial \theta} (\theta^T A \theta) = 2A\theta$$

  Where A is a symmetric matrix, and θ is a vector. This result is commonly used in the derivation of Ridge Regression.

7.2 Common Matrix Derivative Results

- Derivative of a scalar with respect to a vector:

  $$\frac{\partial}{\partial \theta} \mathbf{y}^T \theta = \mathbf{y}$$

  Where $\mathbf{y}$ is a vector and θ is the vector of parameters.

- Derivative of the sum of squared errors:

  $$\frac{\partial}{\partial \theta}\left(\frac{1}{2}(\mathbf{X}\theta - \mathbf{y})^T(\mathbf{X}\theta - \mathbf{y})\right) = \mathbf{X}^T(\mathbf{X}\theta - \mathbf{y})$$

  This formula is used extensively in linear regression, Ridge regression, and Elastic Net.

- Derivative of the Frobenius norm (L2 norm):

  $$\frac{\partial}{\partial \theta}\|\theta\|_2^2 = 2\theta$$

  This is the basis for the penalty term in Ridge Regression.

- Derivative of the L1 norm:

  $$\frac{\partial}{\partial \theta}\|\theta\|_1 = \text{sign}(\theta)$$

  The sign function $\text{sign}(\theta)$ returns 1 for positive elements, -1 for negative elements, and 0 for zero elements. This is used in Lasso regression.

These matrix derivative rules are critical for deriving the optimal solutions for linear models and understanding the behavior of regularization methods like Ridge, Lasso, and Elastic Net.