Machine Learning Notes

Amit Thakur

January 3, 2025

Contents

Chapter 1

Linear Regression

Linear Regression is a statistical method used for modeling the relationship between a dependent variable (target or output) and one or more independent variables (predictors or features).

1.1 Linear Regression with Single Feature

This regression deals with one independent variable (x).

1.1.1 Linear Regression Model

$$y = \theta_0^t + \theta_1^t x + \epsilon \tag{1.1}$$

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x \tag{1.2}$$

where:

- y: the dependent variable (target)
- x: the independent variable or input feature used to predict y.
- θ_1^t: the slope or coefficient of x at iteration t
- θ_0^t: intercept at iteration t
- ϵ: the error term or residual. It captures the noise or other unmodeled effects.
- $\hat{y}$: the output of the linear regression model () for a given input (x)
- $h_\theta(x)$: the hypothesis function for linear regression.

1.1.2 Cost Function

The cost function for linear regression measures the average squared error between predicted and actual values. It is defined as:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)^2 = \frac{1}{2m} \sum_{i=1}^{m} \left((\theta_0 + \theta_1 x^{(i)}) - y^{(i)}\right)^2 \tag{1.3}$$

where:

- m: Number of input and output datapoints.
- $\frac{1}{2}$: A factor kept for convenience, as it simplifies the derivative calculations during gradient descent.

1.1.3 Minimizing Cost

The error for each data point can be written as:

$$e^{(i)} = y^{(i)} - (\theta_0 + \theta_1 x^{(i)}) \tag{1.4}$$

The cost function becomes:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} e^{(i)^2} \tag{1.5}$$

We need to find θ_0 and θ_1 that minimize J. This requires setting the partial derivatives of J with respect to θ_0 and θ_1 to zero as the cost function is quadratic and there's just one critical point.

Solving for θ_0:

$$\frac{\partial J}{\partial \theta_0} = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - (\theta_0 + \theta_1 x^{(i)}) \right) = 0$$

$$\sum_{i=1}^{m} \left(y^{(i)} - (\theta_0 + \theta_1 x^{(i)}) \right) = 0$$
$$\sum_{i=1}^{m} y^{(i)} = m\theta_0 + \theta_1 \sum_{i=1}^{m} x^{(i)}$$

Divide through by m:

$$\bar{y} = \theta_0 + \theta_1 \bar{x}$$

where $\bar{y}$ and $\bar{x}$ are the means of $y^{(i)}$ and $x^{(i)}$, respectively.
Rearranging:

$$\theta_0 = \bar{y} - \theta_1 \bar{x} \tag{1.6}$$

Solving for θ_1:

$$\frac{\partial J}{\partial \theta_1} = -\frac{1}{m} \sum_{i=1}^{m} x^{(i)} \left(y^{(i)} - (\theta_0 + \theta_1 x^{(i)}) \right) = 0$$

Expanding:

$$\sum_{i=1}^{m} x^{(i)} y^{(i)} = \theta_0 \sum_{i=1}^{m} x^{(i)} + \theta_1 \sum_{i=1}^{m} \left(x^{(i)} \right)^2$$

Substitute $\theta_0 = \bar{y} - \theta_1\bar{x}$:

$$\sum_{i=1}^{m} x^{(i)}y^{(i)} = (\bar{y} - \theta_1\bar{x})\sum_{i=1}^{m} x^{(i)} + \theta_1\sum_{i=1}^{m}(x^{(i)})^2$$

Simplify:

$$\sum_{i=1}^{m} x^{(i)}y^{(i)} = \bar{y}\sum_{i=1}^{m} x^{(i)} - \theta_1\bar{x}\sum_{i=1}^{m} x^{(i)} + \theta_1\sum_{i=1}^{m}(x^{(i)})^2$$

Reorganize terms:

$$\theta_1\left(\sum_{i=1}^{m}(x^{(i)})^2 - \frac{1}{m}\left(\sum_{i=1}^{m} x^{(i)}\right)^2\right) = \sum_{i=1}^{m} x^{(i)}y^{(i)} - \frac{1}{m}\sum_{i=1}^{m} x^{(i)}\sum_{i=1}^{m} y^{(i)}$$

Using simplified notation:

- $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$ (mean of x)
- $\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y^{(i)}$ (mean of y)

$$\theta_1 = \frac{\sum_{i=1}^{m}\left(x^{(i)} - \bar{x}\right)\left(y^{(i)} - \bar{y}\right)}{\sum_{i=1}^{m}\left(x^{(i)} - \bar{x}\right)^2} \tag{1.7}$$

1.1.4 Gradient Descent Algorithm to find optimal θ_0 and θ_1

1. Gradient descent moves the parameters in the direction of the negative gradient (steepest descent) of the cost function.
2. The learning rate α determines the size of each step. If α is too large, the algorithm may overshoot the minimum. If it is too small, convergence may take too long.
3. The iterative process ensures gradual improvement in the model parameters until the cost function is minimized.

Algorithm 1 Gradient Descent for Linear Regression

```
1: **Input:** Learning rate $\alpha$, initial values for $\theta_0$ and $\theta_1$, and maximum iterations or convergence threshold $\epsilon$.
2: **Output:** Optimized parameters $\theta_0$ and $\theta_1$.
3: Set initial values for $\theta_0$ and $\theta_1$.
4: **repeat**
5:     Compute updates for parameters:
           $$\text{temp0} := \theta_0 - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)$$
           $$\text{temp1} := \theta_1 - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right) x^{(i)}$$
6:     Update parameters simultaneously:
           $$\theta_0 := \text{temp0}$$
           $$\theta_1 := \text{temp1}$$
7:     Compute cost function:
           $$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$
8: **until** maximum iterations reached **or** change in $J(\theta_0, \theta_1)$ is below threshold $\epsilon$.
9: **return** $\theta_0, \theta_1$
```

1.2 Linear Regression with Multiple Features

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n + e \tag{1.8}$$

where

- y: The dependent variable (or target/output variable) that the model predicts
- θ_0: The intercept term, representing the value of y when all $x_i = 0$
- $\theta_1, \theta_2, \ldots, \theta_n$: The coefficients or weights for the independent variables $x_1, x_2, \ldots, x_n$.

1.2.1 General Form

$$y = \boldsymbol{\theta}^T \mathbf{x} + e \tag{1.9}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{(n+1)\times 1}$ is the parameter vector (column vector of coefficients), defined as:

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \tag{1.10}$$

And the input feature vector $\mathbf{x} \in \mathbb{R}^{(n+1)\times 1}$ defined as:

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{1.11}$$

The prediction function is:

$$\hat{y} = h_\theta(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} \tag{1.12}$$

1.2.2 Cost Function

The residuals (E) represent the difference between the actual target values (y) and the predicted values ($X \cdot \boldsymbol{\theta}$). For m data points:

$$E = \begin{bmatrix} e^{(1)} \\ e^{(2)} \\ \vdots \\ e^{(m)} \end{bmatrix} = X \cdot \boldsymbol{\theta} - y \tag{1.13}$$

The cost function $J(\boldsymbol{\theta})$ is:

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} \left(e^{(i)}\right)^2 \tag{1.14}$$

Using $E = X \cdot \boldsymbol{\theta} - y$, we can write:

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \|E\|^2$$

Expanding this using the transpose:

$$J(\boldsymbol{\theta}) = \frac{1}{2m} E^T E$$

Substituting $E = X \cdot \boldsymbol{\theta} - y$:

$$J(\boldsymbol{\theta}) = \frac{1}{2m} (X \cdot \boldsymbol{\theta} - y)^T (X \cdot \boldsymbol{\theta} - y) \tag{1.15}$$

To minimize $J(\boldsymbol{\theta})$, we compute the gradient (partial derivative) with respect to each parameter θ_j:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} \tag{1.16}$$

We now compute the partial derivative of $J(\boldsymbol{\theta})$ with respect to θ_j. The chain rule will be used.

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \left[(X \cdot \boldsymbol{\theta})^T (X \cdot \boldsymbol{\theta}) - 2y^T (X \cdot \boldsymbol{\theta}) + y^T y\right]$$

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \left[\boldsymbol{\theta}^T X^T X \boldsymbol{\theta} - 2y^T X \boldsymbol{\theta}\right] + \text{constant terms independent of } \boldsymbol{\theta}$$

Compute the derivative with respect to $\boldsymbol{\theta}$ (matrix calculus):

- Derivative of $\boldsymbol{\theta}^T X^T X \boldsymbol{\theta}$:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \left(\boldsymbol{\theta}^T X^T X \boldsymbol{\theta} \right) = 2X^T X \boldsymbol{\theta}$$

- Derivative of $-2y^T X \boldsymbol{\theta}$:

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

Substituting above values, we get:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{m} \left(X^T X \boldsymbol{\theta} - X^T y \right) \tag{1.17}$$

From the general gradient, we can isolate the derivative with respect to a single parameter θ_j. The gradient is:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{m} X^T (X \cdot \boldsymbol{\theta} - y)$$

The j-th element of the gradient corresponds to:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{m} (X_{\text{column},j})^T (X \cdot \boldsymbol{\theta} - y)$$

1.2.3 Gradient of the Cost Function

The gradient of the cost function with respect to all parameters $\boldsymbol{\theta}$ is a vector:

$$\text{grad} = \begin{bmatrix} \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_0} \\ \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_n} \end{bmatrix} \tag{1.18}$$

$$\text{grad} = \frac{1}{m} X^T (X \cdot \boldsymbol{\theta} - y) \tag{1.19}$$

1.2.4 Analytical Solution

To find the optimal $\boldsymbol{\theta}$, set the gradient grad to zero:

$$\frac{1}{m} X^T (X \cdot \boldsymbol{\theta} - y) = 0$$

Simplify:

$$X^T (X \cdot \boldsymbol{\theta}) = X^T y$$

Rearranging:

$$\boldsymbol{\theta} = (X^T X)^{-1} X^T y \tag{1.20}$$

This is known as the normal equation, which gives the closed-form solution for $\boldsymbol{\theta}$ in linear regression.

1.2.5 Gradient Descent Algorithm

Algorithm 2 Gradient Descent for Linear Regression for multiple features

```
1: Input: Learning rate $\alpha$, initial values for $\boldsymbol{\theta}$, and maximum iterations or convergence
   threshold $\epsilon$.
2: Output: Optimized parameter vector, $\boldsymbol{\theta}$.
3: repeat
4:     Compute gradient vector: $\mathbf{grad} \in \mathbb{R}^{(n+1)\times 1}$:
           $$\mathbf{grad} = X^T(X \cdot \boldsymbol{\theta} - y)$$
5:     Update the parameter vector, $\boldsymbol{\theta}$:
           $$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \cdot \mathbf{grad}$$
6:     Compute cost function (just for monitoring):
           $$J(\boldsymbol{\theta}) = \frac{1}{2m}(X \cdot \boldsymbol{\theta} - y)^T(X \cdot \boldsymbol{\theta} - y)$$
7: until maximum iterations reached or change in $J(\boldsymbol{\theta})$ is below threshold $\epsilon$.
8: return $\boldsymbol{\theta}$
```

Bibliography

[1] Weidong Kuang. *Fundamentals of deep learning: a step-by-step guide.* Self, 2024. `https://faculty.utrgv.edu/weidong.kuang/book/book.html`(visited 2024-12-01).

[2] Simon J.D. Prince. *Understanding Deep Learning.* The MIT Press, 2023.

[3] Amit Thakur. *Maths for Machine Learning Notes.* Self, 2025. `http://amitthakur.org/learnings/maths/maths-for-ml`.