Multiple Linear Regression

Derivation and Gradient Descent Optimization

Kiran Bagale

Supervised Learning

July, 2024

Outline

Multiple Linear Regression Problem

Real-World Example

Predicting house prices in Kathmandu based on multiple features:

- Number of bedrooms
- Location
- Number of floors
- Area size
- Other relevant features

Mathematical Representation

We have m observations with n features each:

- Independent variables: $x_1, x_2, \ldots, x_n$
- Dependent variable: y
- Dataset: $(x^{(i)}, y^{(i)})$ where $i = 1, 2, \ldots, m$

The Linear Model

Linear Relationship

The relationship between features and target can be expressed as:

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

Prediction Function

For predictions, we use:

$$\hat{y} = h(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

where:

- $\hat{y} = h(x)$ is the predicted value
- w_0 is the bias term (intercept)
- w_j is the coefficient of the j-th feature
- x_j is the j-th independent variable

Residuals and Errors

Definition of Residuals

The difference between actual and predicted values:

$$e = \hat{y} - y$$

Total Error

For all observations:

$$\text{Total Error} = \sum_{i=1}^{m} e^{(i)}$$

Problem with Simple Sum

Positive and negative errors can cancel each other out, giving misleading results!

Error Measures

Total Absolute Error

To avoid cancellation:

$$\text{Total Error} = \sum_{i=1}^{m} |e^{(i)}|$$

Residual Sum of Squares (RSS)

Alternatively, square the errors:

$$RSS = \sum_{i=1}^{m}(e^{(i)})^2 = \sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$$

Mean Squared Error (MSE)

Average of squared errors:

Cost Function

MSE Cost Function

We define our cost function as:

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

- $h(x^{(i)})$ is the predicted value for the i-th example
- $y^{(i)}$ is the actual value for the i-th example
- The factor $\frac{1}{2}$ is for mathematical convenience in derivatives

Goal

Minimize $J(w_j)$ to find optimal parameters w_j

Gradient Descent Algorithm

Optimization Technique

Gradient Descent is an iterative algorithm that finds the minimum of a function by:

- Taking steps proportional to the negative gradient
- Moving towards the steepest descent direction

Update Rule

$$w_j := w_j - \alpha \frac{\partial J(w)}{\partial w_j}$$

where:

- α is the learning rate
- $\frac{\partial J(w)}{\partial w_j}$ is the partial derivative of cost function

Derivative Calculation

For Single Training Example

$$\frac{\partial}{\partial w_j}\frac{1}{2}(h(x)-y)^2$$

Step-by-Step Derivation

$$= \frac{1}{2}\cdot 2(h(x)-y)\cdot\frac{\partial}{\partial w_j}(h(x)-y) \quad (1)$$

$$= (h(x)-y)\cdot\frac{\partial}{\partial w_j}(w_0 + w_1x_1 + \cdots + w_nx_n) \quad (2)$$

$$= (h(x)-y)\cdot x_j \quad (3)$$

LMS Update Rule (Widrow-Hoff)

$$w_j := w_j - \alpha(h(x)-y)\cdot x_j$$

Stochastic Gradient Descent (SGD)

Characteristics

- Uses only **one** training example at a time
- Faster per iteration
- Requires less memory
- Produces noisier gradients
- Can escape local minima

Update Rule

$$w_j := w_j + \alpha(y^{(i)} - h(x^{(i)})) \cdot x_j^{(i)}$$

Advantages

- Efficient for large datasets
- Faster convergence in practice
- Better for online learning

Stochastic Gradient Descent Algorithm

Algorithm 1 Stochastic Linear Regression

Require: Training data X, targets y, learning rate α, epochs n
Ensure: Optimized weights w

Initialize weights $w \leftarrow$ random small values
for epoch $= 1$ to n **do**
    Shuffle training data (X, y)
    for each (x_i, y_i) in (X, y) **do**
        prediction $\leftarrow w^T \cdot x_i$
        error $\leftarrow$ prediction $-y_i$
        gradient $\leftarrow$ error $\cdot x_i$
        $w \leftarrow w - \alpha \cdot$ gradient
    end for
end for
return w

Batch Gradient Descent

Characteristics

- Uses the **entire** training dataset
- More stable gradients
- Guaranteed convergence to global minimum
- Computationally expensive for large datasets

Update Rule

$$w_j := w_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h(x^{(i)})) \cdot x_j^{(i)}$$

Gradient Formulas

$$\frac{\partial J}{\partial w_0} = -\frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)}) \tag{4}$$

Batch Gradient Descent Algorithm

Algorithm 2 Batch Linear Regression

Require: Training data X, targets y, learning rate α, iterations n

Ensure: Optimized weights w

```
Initialize weights w ← 0
m ← number of training examples
for i = 1 to n do
    predictions ← X · w
    errors ← predictions − y
    gradient ← (1/m) X^T · errors
    w ← w − α · gradient
end for
return w
```

Example Dataset

Dataset

Observation	x_1	x_2	y
1	1	2	4
2	2	3	7
3	3	4	10

Hypothesis Function

$$h(x) = w_0 + w_1 x_1 + w_2 x_2$$

Initial Parameters

$$w_0 = 0, \quad w_1 = 0, \quad w_2 = 0, \quad \alpha = 0.01$$

Iteration 1

Predictions

$$h(1,2) = 0 + 0 \cdot 1 + 0 \cdot 2 = 0 \quad (6)$$
$$h(2,3) = 0 + 0 \cdot 2 + 0 \cdot 3 = 0 \quad (7)$$
$$h(3,4) = 0 + 0 \cdot 3 + 0 \cdot 4 = 0 \quad (8)$$

Gradients

$$\frac{\partial J}{\partial w_0} = -\frac{1}{3}[(0-4) + (0-7) + (0-10)] = 7 \quad (9)$$
$$\frac{\partial J}{\partial w_1} = -\frac{1}{3}[(0-4) \cdot 1 + (0-7) \cdot 2 + (0-10) \cdot 3] = 5.33 \quad (10)$$
$$\frac{\partial J}{\partial w_2} = -\frac{1}{3}[(0-4) \cdot 2 + (0-7) \cdot 3 + (0-10) \cdot 4] = 7.67 \quad (11)$$

Parameter Updates

Iteration 2

Updated Predictions

$$h(1,2) = -0.07 - 0.0533 \cdot 1 - 0.0767 \cdot 2 = -0.277 \tag{15}$$
$$h(2,3) = -0.07 - 0.0533 \cdot 2 - 0.0767 \cdot 3 = -0.432 \tag{16}$$
$$h(3,4) = -0.07 - 0.0533 \cdot 3 - 0.0767 \cdot 4 = -0.589 \tag{17}$$

New Parameter Values

$$w_0 = -0.140 \tag{18}$$
$$w_1 = -0.107 \tag{19}$$
$$w_2 = -0.153 \tag{20}$$

Convergence

The algorithm continues iterating until convergence criteria are met!

Key Takeaways

Multiple Linear Regression

- Models linear relationships between multiple features and target
- Uses MSE as cost function to measure prediction accuracy
- Optimized using gradient descent algorithms

Gradient Descent Variants

- **Stochastic**: Fast, noisy, good for large datasets
- **Batch**: Stable, expensive, guaranteed convergence

Applications

- House price prediction
- Stock market analysis
- Sales forecasting
- Any continuous prediction problem

Thank You

Questions?