Supervised Machine Learning: Regression and Classification

***Abstract* — Supervised learning is a core methodology in machine learning that involves training models to map input data to corresponding outputs based on labeled examples. This paper provides an in-depth exploration of supervised learning techniques, focusing on both regression and classification tasks. We begin by discussing the distinction between supervised and unsupervised learning, followed by a detailed explanation of simple and multiple linear regression. The mathematical framework of gradient descent optimization is explored for training models, along with practical implementations of regression with multiple input variables. The discussion then shifts to logistic regression, a widely used method for binary classification, covering its hypothesis function, cost function, and optimization via gradient descent. The issue of overfitting is addressed, and techniques like regularization are introduced to mitigate this challenge. This paper highlights the theoretical underpinnings of supervised learning models while emphasizing their practical significance in real-world applications.**

***Keywords* — Supervised learning, linear regression, multiple linear regression, gradient descent, logistic regression, classification, cost function, overfitting, regularization, machine learning**

I. Introduction to Machine Learning

A. Supervised vs Unsupervised Learning

In supervised learning, we have a dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, where each input $x^{(i)} \in \mathbb{R}^n$ is paired with an output $y^{(i)} \in \mathbb{R}$ (for regression) or $y^{(i)} \in \{0, 1\}$ (for binary classification). The goal is to learn a function $h_\theta(x)$ (called the hypothesis) that maps input features to outputs such that:

$$h_\theta(x) \approx y \text{ for all } (x, y) \in \mathcal{D}.$$

In contrast, in unsupervised learning, we are given only inputs $\{x^{(i)}\}_{i=1}^{m}$ without any labels, and the goal is to learn patterns or structures from the data.

B. Regression Model

In simple linear regression, the hypothesis $h_\theta(x)$ is a linear function of the input:

$$h_\theta(x) = \theta_0 + \theta_1 x.$$

For multiple inputs (features), this extends to:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \theta^T x,$$

where $\theta = [\theta_0, \theta_1, \ldots, \theta_n]^T$ is the vector of model parameters, and $x = [1, x_1, x_2, \ldots, x_n]^T$ is the feature vector (including a constant term for the bias θ_0).

C. Train the Model with Gradient Descent

The cost function for linear regression is the Mean Squared Error (MSE), given by:

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

To minimize this cost function, we use gradient descent, an iterative optimization algorithm that updates the parameters θ in the opposite direction of the gradient of the cost function with respect to θ:

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

where α is the learning rate. The partial derivative of the cost function with respect to θ_j is:

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

Thus, the update rule becomes:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}.$$

II. Regression with Multiple Input Variables

A. Multiple Linear Regression

For multiple input variables, the hypothesis function is:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n.$$

In vectorized form, this is written as:

$$h_\theta(x) = \theta^T x,$$

where $\theta = [\theta_0, \theta_1, \ldots, \theta_n]^T$ and $x = [1, x_1, x_2, \ldots, x_n]^T$.

The cost function remains the same:

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

B. Gradient Descent in Practice

The update rule for gradient descent in multiple linear regression is:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

for $j = 0, 1, 2, \ldots, n$.

In vectorized form, this can be written as:

$$\theta := \theta - \alpha \frac{1}{m} X^T (X\theta - y),$$

where:

- $X \in \mathbb{R}^{m \times (n+1)}$ is the matrix of input features (with a column of ones for the intercept term),
- $y \in \mathbb{R}^m$ is the vector of target values.

The gradient descent update is computed by multiplying the matrix of features X with the vector of errors $(X\theta - y)$.

III. Classification

A. Classification with Logistic Regression

Logistic regression is used for binary classification. The hypothesis function for logistic regression is the sigmoid function (also called the logistic function):

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}.$$

The output $h_\theta(x)$ represents the probability that the input x belongs to the positive class $(y = 1)$:

$$P(y = 1 \mid x; \theta) = h_\theta(x).$$

B. Cost Function for Logistic Regression

The cost function for logistic regression is derived from the log-likelihood of the Bernoulli distribution. The cost function (also called the binary cross-entropy or log-loss) is:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(x^{(i)})) \right] + \left[(1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

This function penalizes both false positives and false negatives, and it is convex, ensuring that gradient descent will converge to the global minimum.

C. Gradient Descent for Logistic Regression

To minimize the log-loss cost function, we again use gradient descent. The partial derivative of the cost function with respect to θ_j is:

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

Thus, the gradient descent update rule for logistic regression is:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \quad \text{for} \quad j = 0, 1, 2, \ldots, n.$$

In vectorized form, the update rule becomes:

$$\theta := \theta - \alpha \frac{1}{m} X^T (h_\theta(X) - y),$$

where $h_\theta(X) = \frac{1}{1 + e^{-X\theta}}$ is the vector of predicted probabilities.

D. The Problem of Overfitting

Overfitting occurs when the model becomes too complex and fits the noise in the training data rather than the underlying pattern. To prevent overfitting, we use regularization techniques like L2 regularization (also called Ridge Regression). The regularized cost function for logistic regression is:

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

where λ is the regularization parameter. The term

$$\frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

penalizes large weights, helping to prevent the model from overfitting by constraining the parameter values.

The gradient descent update rule for the regularized cost function becomes:

$$\theta_j := \theta_j - \alpha \left(\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right) \quad \text{for} \quad j = 1, 2, \ldots, n,$$

with no regularization on the intercept term θ_0.