Mahcine Learning Notes

Ahmed Yasser

2025

Contents

Some Important Definitions

- θ = Parameters of the model
- X = Inputs or features
- y = Output or target variable
- $h(x)$ = Hypothesis function
- m = Number of training examples (Number of rows in the training set)
- n = Number of features
- $\left(X^{(i)}, y^{(i)}\right)$ = The $i\text{-}th$ training example

Supervised Learning

Supervised learning is a type of machine learning where the model is trained on labeled data. The goal is to learn a mapping from inputs to outputs based on the provided labels. The model can then make predictions on new, unseen data.

Supervised Learning Types

Supervised learning can be divided into two main types:

1. **Regression**: Predicts continuous numerical values or quantities within a range, such as prices, temperatures, or heights.
2. **Classification**: Predicts discrete categories or classes, assigning input data to predefined groups such as spam/not spam or specific object types.

Some common supervised learning algorithms that can only be used for regression tasks include:

- Linear Regression
- Polynomial Regression
- Support Vector Regression (SVR)
- Gaussian Processes Regression
- Rebust Regression

Some common supervised learning algorithms that can only be used for classification tasks include:

- Logistic Regression (and its extensions)
- Support Vector Machines (SVM)
- Nearest Neighbors (KNN)
- Linear Discriminant Analysis (LDA)

Some common supervised learning algorithms that can be used for both regression and classification tasks include:

- Decision Trees
- Random Forests
- Gradient Boosting Machines (GBM)
- Ridge Regression
- Lasso Regression
- AdaBoost

Linear Regression

Linear regression is a supervised learning algorithm used for regression tasks. It models the relationship between input features and a continuous output variable by fitting a linear equation to the observed data. The goal is to find the best-fitting line (or hyperplane in higher dimensions) that minimizes the difference between the predicted values and the actual values.

Linear Regression Equation

The hypothesis function for linear regression is represented as:

Hypothesis Function for Linear Regression

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n = \sum_{j=0}^{n} \theta_j x_i$$

Where:

- $h(x)$ or $h_\theta(x)$ is the predicted output (hypothesis function)
- θ_0 is the bias
- $\theta_1, \theta_2, ..., \theta_n$ are the weights (coefficients) for each feature
- $x_1, x_2, ..., x_n$ are the input features

The equation can also be expressed in vector form as:

Vector Form of Linear Regression Hypothesis Function

$$h(x) = \theta^T X$$

Where:

$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ ... \\ \theta_n \end{pmatrix}$ is the vector of parameters (weights)

$X = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ ... \\ x_n \end{pmatrix}$ is the vector of input features.

Cost Function

The cost function for linear regression is used to measure how well the model fits the training data. It calculates the difference between the predicted values and the actual values. The most common cost function used in linear regression is the Mean Squared Error (MSE):

Cost Function for Linear Regression

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

Where:

- $J(\theta)$ is the cost Function
- m is the number of training examples
- $h\left(x^{(i)}\right)$ is the predicted value for the $i\text{-}th$ training examples
- $y^{(i)}$ is the actual value for the $i\text{-}th$ training examples
- θ is the vector of parameters (weights)
- $x^{(i)}$ is the input features for the $i\text{-}th$ training examples

Batch Gradient Descent

Batch gradient descent is an optimization algorithm used to minimize the cost function by iteratively updating the model parameters (weights). It computes the gradient of the cost function with respect to the parameters for the entire training dataset and updates the parameters in the opposite direction of the gradient.

Batch Gradient Descent Algorithm

Repeat until convergence:

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

Where:

- $j = 0, 1, ..., n$ (for all features)
- θ_j is the parameter (weight) for the $j\text{-}th$ feature
- α is the learning rate (step size)
- $\frac{\partial}{\partial \theta_j} J(\theta)$ is the partial derivative of the cost function with respect to the parameter θ_j
- $x_j^{(i)}$ is the value of the $j\text{-}th$ feature for the $i\text{-}th$ training example

Stochastic Gradient Descent (SGD)

Stochastic gradient descent (SGD) is a variant of gradient descent that updates the model parameters using only one training example at a time. This can lead to faster convergence.

Stochastic Gradient Descent Algorithm

Repeat:

For $i = 1$ to m:

$$\theta_j := \theta_j - \alpha\big(h\big(x^{(i)}\big) - y^{(i)}\big)x_j^{(i)} \qquad \text{For } j = 0 \text{ to } n$$

Where:

- θ_j is the parameter (weight) for the $j\text{-}th$ feature
- α is the learning rate (step size)
- $h\big(x^{(i)}\big)$ is the predicted value for the $i\text{-}th$ training example
- $y^{(i)}$ is the actual value for the $i\text{-}th$ training example
- $x_j^{(i)}$ is the value of the $j\text{-}th$ feature for the $i\text{-}th$ training example