

# Basic Machine Learning Algorithms

Nguyen Thai Son  
 Faculty of Computer Science  
 Hanoi, Vietnam  
 Email: titque1107@gmail.com

**Abstract**—This paper presents four basic Machine Learning algorithms: Linear Regression, K-Means, Gradient Descent, and K-Nearest Neighbors (KNN). Each algorithm is introduced with its mathematical principle, illustration, and typical applications in real-world problems.

## I. LINEAR REGRESSION

### A. Introduction

Linear Regression is one of the most fundamental algorithms in Machine Learning, used to predict a continuous value based on one or more input features.

### B. Mathematics analysis

For a single variable:

$$\hat{y} = w_0 + w_1 x$$

where:

- $\hat{y}$ : predicted value
- $x$ : input variable
- $w_1$ : coefficient
- $w_0$ : intercept (the value of  $y$  when  $x = 0$ )

For multiple variables:

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

This is called **multiple linear regression**.

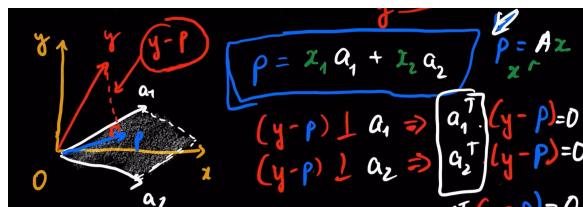


Fig. 1: Behind maths explanation

Final mathematics:

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

where:

- $\mathbf{X}$  is the  $m \times n$  matrix of input features.
- $\mathbf{y}$  is the  $m \times 1$  vector of target values.
- $\mathbf{w}$  is the  $n \times 1$  vector of model parameters (weights).

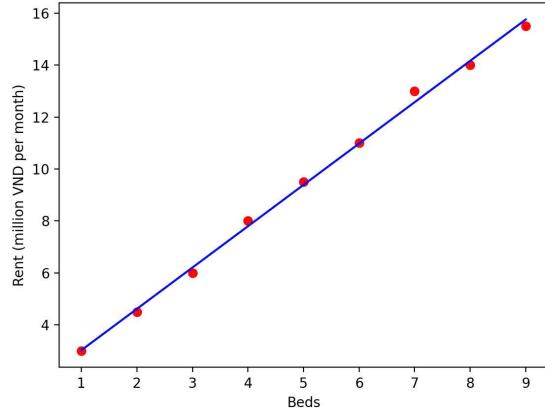


Fig. 2: Example of a 2D fitted line in Linear Regression.

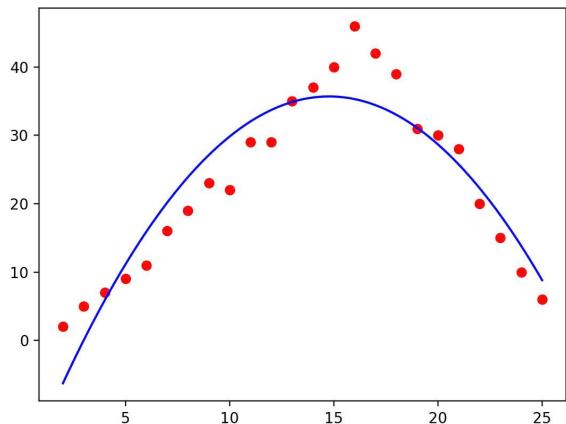


Fig. 3: Example of a 2D fitted parabol in Linear Regression.

### C. Applications

Linear Regression is widely used for predicting continuous outcomes such as house prices, sales forecasts, and trend analysis. Some common applications include:

- **Prediction:** Estimate continuous values (e.g., house price, sales, marks).
- **Trend Analysis:** Study time-based trends (e.g., GDP, temperature).
- **Economics/Finance:** Model relationships (e.g., income vs. spending, CAPM).
- **Marketing:** Measure advertising impact and predict sales.

## II. K-MEANS CLUSTERING

### A. Introduction

K-Means is one of the most popular unsupervised learning algorithms used to partition a dataset into  $k$  distinct clusters based on feature similarity. Each data point is assigned to the cluster with the nearest centroid, and centroids are updated iteratively to minimize the overall within-cluster variance.

For example, consider a dataset of houses with attributes such as size, number of floors, and location. K-Means can group them into categories such as luxury, average, or low-cost based on these numerical features (after vectorization).

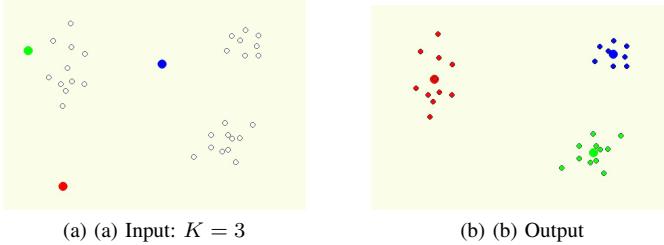


Fig. 4: Example of K-Means clustering on 2D points.

### B. Mathematical Analysis

The K-Means algorithm takes a set of observations

$$X = [x_1, x_2, \dots, x_N] \in \mathbb{R}^{d \times N},$$

where each data point  $x_i \in \mathbb{R}^d$  represents a vector in  $d$ -dimensional space.

The algorithm aims to minimize the objective function:

$$J = \sum_{i=1}^N \sum_{j=1}^k y_{ij} \|x_i - \mu_j\|^2,$$

where  $\mu_j$  is the centroid of cluster  $j$ , and  $y_{ij} = 1$  if point  $x_i$  belongs to cluster  $j$ , otherwise 0.

### Notation

- $N$ : Number of observations.
- $k$ : Number of clusters.
- $x_i$ :  $i$ -th data point.
- $\mu_j$ : Centroid of cluster  $j$ .
- $y_{ij}$ : Labels for cluster assignment.

### C. Algorithm Summary and Flowchart

1) 1) **Summary:** The K-Means algorithm can be summarized as an iterative process of alternately fixing the cluster assignments  $Y$  and the cluster centroids  $M$  until convergence. The main steps are as follows:

- 1) **Step 1:** Specify the number of clusters  $k$ .
- 2) **Step 2:** Randomly initialize  $k$  points as the initial cluster centroids.
- 3) **Step 3:** Assign each data point to the cluster with the nearest centroid based on the Euclidean distance.

- 4) **Step 4:** Update each centroid as the mean of all data points assigned to its cluster.
- 5) **Step 5:** Repeat Steps 3 and 4 until the centroids no longer change (i.e., the algorithm converges).

2) 2) **Flowchart:** The following diagram illustrates the workflow of the K-Means algorithm.

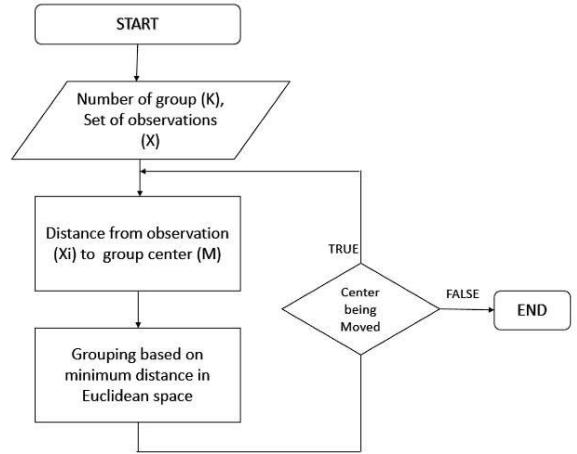


Fig. 5: Flowchart of the K-Means clustering algorithm.

### D. Applications

K-Means is applied to reduce image size by decreasing the number of unique colors. Each pixel (an observation with RGB values) is clustered into one of  $K$  color groups. Pixels are then reassigned to the nearest cluster centroid, creating a compressed image with fewer color shades. The experiment is implemented in Pascal using the SwinGame API.

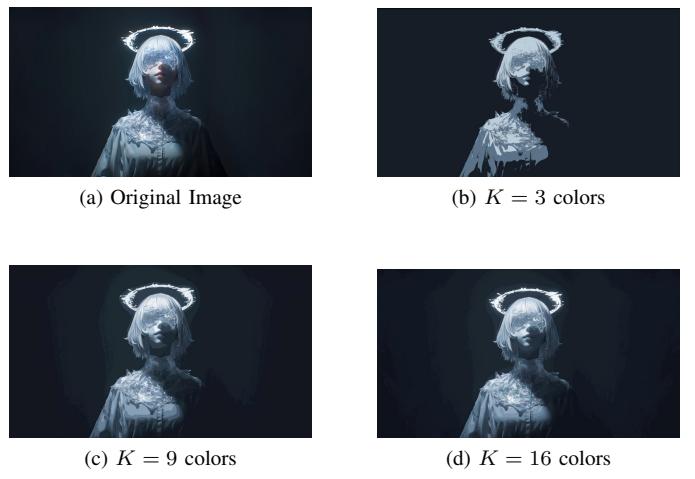


Fig. 6: Image compression using K-Means algorithm with different numbers of color clusters.

### III. GRADIENT DESCENT

#### A. Introduction

Gradient Descent is an optimization algorithm used to minimize a cost function, scaled by a factor called the **learning rate**  $\alpha$ .

#### B. Mathematics analysis

##### Algorithm Steps:

We define the cost function as:

$$f(x) = \|Ax - y\|^2$$

which can be expanded as:

$$f(x) = (Ax - y)^T(Ax - y)$$

Simplifying, the gradient can be written as:

$$f'(x) = 2A^T(Ax - y)$$

This gradient is used in the Gradient Descent update rule:

$$x := x - \alpha f'(x)$$

where  $\alpha$  is the learning rate.

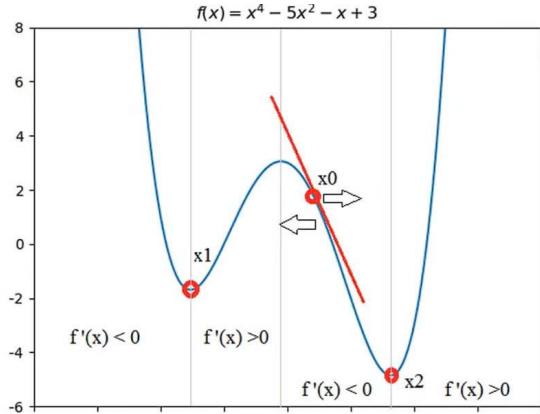


Fig. 7: Example of Gradient Descent Algorithm.

The first plot illustrates how the regression line gradually adjusts its slope and intercept at each iteration, moving closer to the optimal solution.

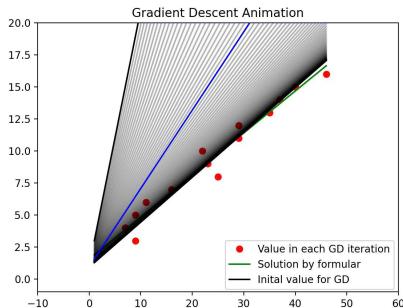


Fig. 8: Gradient Descent Iterations. The plot illustrates how the regression line gradually adjusts its slope and intercept at each iteration, moving closer to the optimal solution.

The second plot shows the convergence of the cost function over iterations, demonstrating how the error value decreases steadily as the algorithm approaches the minimum.

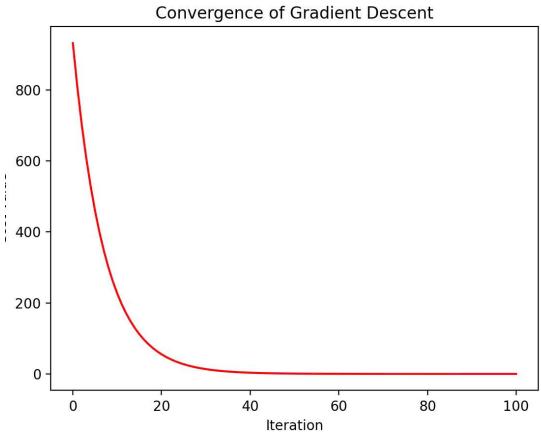


Fig. 9: Cost Function Convergence. The plot shows how the error value decreases steadily as the algorithm approaches the minimum.

This algorithm can also be visualized as moving along a parabolic cost function in one dimension. However, in practice, it can be extended to much higher dimensions, where the cost function  $J(\theta)$  depends on multiple parameters. In these higher-dimensional spaces, the algorithm follows the negative gradient direction through a complex cost surface to reach the global or local minimum.

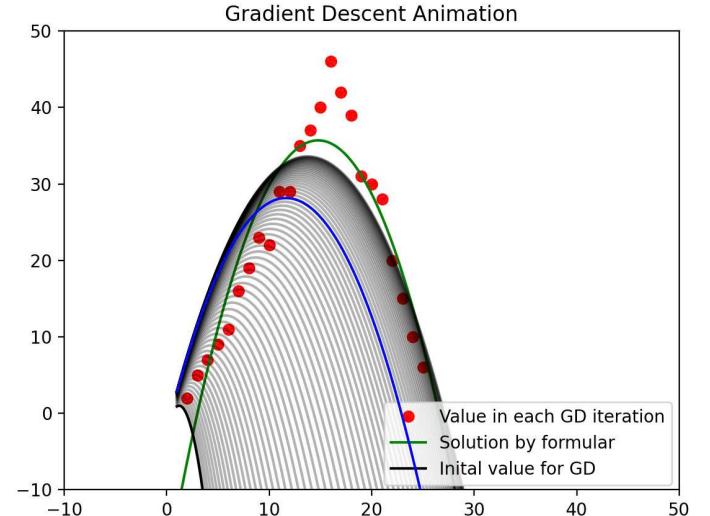


Fig. 10: Visualization of Gradient Descent on a parabolic cost function (1D). The same principle extends to higher-dimensional cost surfaces.

#### C. Applications

It is commonly used in training machine learning models, including linear regression and deep neural networks.

## IV. K-NEAREST NEIGHBORS (KNN)

### A. Introduction

K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm used for both classification and regression tasks. It operates under the assumption that similar points exist close to each other in feature space — in other words, “birds of a feather flock together”. When a new observation is given, KNN looks at the  $k$  closest data points in the training set and makes predictions based on their labels or values.

### B. Mathematical Analysis

Given a dataset:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\},$$

where  $x_i \in \mathbb{R}^d$  is a feature vector and  $y_i$  is its corresponding label.

For a new point  $x_{\text{new}}$ , the algorithm computes its distance to all training samples, commonly using the Euclidean distance:

$$d(x_{\text{new}}, x_i) = \sqrt{\sum_{j=1}^d (x_{\text{new},j} - x_{i,j})^2}.$$

After sorting distances in ascending order, the algorithm selects the  $k$  nearest neighbors.

### C. Algorithm Steps

- 1) **Step 1:** Choose the number of neighbors  $k$ .
- 2) **Step 2:** Compute the distance between the test point and all training samples.
- 3) **Step 3:** Sort all distances and select the  $k$  nearest neighbors.
- 4) **Step 4:** For classification, count the majority label; for regression, compute the average value.
- 5) **Step 5:** Return the predicted output  $\hat{y}$ .

### D. Example Visualization

1) **Example 1: Iris Flower Classification:** The **Iris dataset** contains 150 samples from three species of iris flowers (*Setosa*, *Versicolor*, and *Virginica*). Each sample is described by four numerical features: sepal length, sepal width, petal length, and petal width.

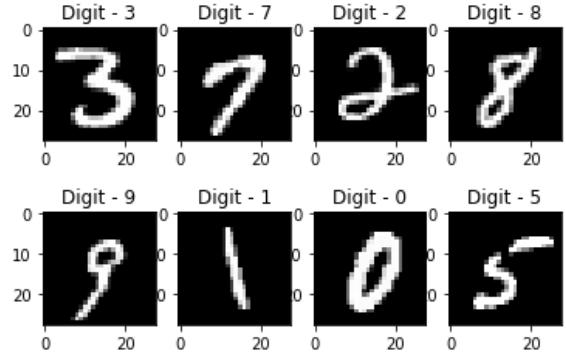
a) **Training and Testing Phase::** The dataset is divided into two subsets — typically 80% for training and 20% for testing. During training, KNN stores all training samples in memory. When a new flower (test sample) is introduced, the algorithm calculates its distance to all training samples and selects the  $k$  nearest ones. The majority class among these neighbors determines the final prediction.

2) **Example 2: Handwritten Digit Classification:** The **handwritten digit recognition** problem (e.g., MNIST or scikit-learn digits dataset) contains images of digits from 0 to 9. Each image is represented as a  $28 \times 28$  grayscale grid, which is flattened into a 784-dimensional feature vector.



Fig. 11: KNN classification on the Iris dataset (2D projection using PCA). Different colors represent flower species, and decision boundaries are learned from the training set.

a) **Training and Testing Phase::** The dataset is split into training and testing sets (e.g., 60,000 for training and 10,000 for testing). During training, all digit images and their labels are stored. When a new image is given, the algorithm measures the Euclidean distance between this image and all training images, finds the  $k$  closest ones, and predicts the most frequent digit among them.



### E. Combined Application of Linear Regression and KNN

In this experiment, both Linear Regression and K-Nearest Neighbors (KNN) are applied to solve the same prediction problem — estimating house prices based on features such as size, number of rooms, and distance to the city center.

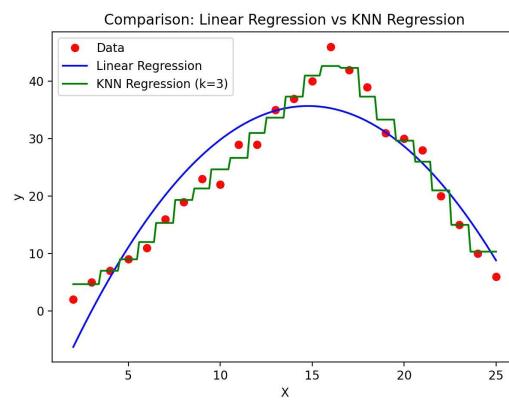


Fig. 12: Comparison between Linear Regression and KNN predictions. The red line represents Linear Regression (global trend), while blue dots show KNN predictions based on local neighborhoods.

TABLE I: Comparison of Linear Regression, K-Means, Gradient Descent, and KNN

Algorithm	Type	Pros	Cons	Complexity	Recommended Use
<b>Linear Regression</b>	Supervised (Regression)	Simple, interpretable, fast to train, low memory usage	Assumes linearity, sensitive to outliers	Low	Predicting continuous outcomes with linear relationships
<b>K-Means</b>	Unsupervised (Clustering)	Scalable, easy to interpret, efficient on large datasets	Requires predefined $K$ , sensitive to initialization, non-convex objective	Medium	Grouping unlabeled data into clusters or compressing image colors
<b>Gradient Descent</b>	Optimization Algorithm	Widely applicable, efficient for differentiable models, scalable to large data	May converge slowly, depends on learning rate, possible local minima	Variable	Training models such as regression, neural networks, or SVMs
<b>K-Nearest Neighbors (KNN)</b>	Supervised (Regression/Classification)	Non-parametric, captures non-linear patterns, simple to understand	Slow at prediction, sensitive to noise and feature scaling	High (during prediction)	Small datasets or problems with irregular decision boundaries