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.**

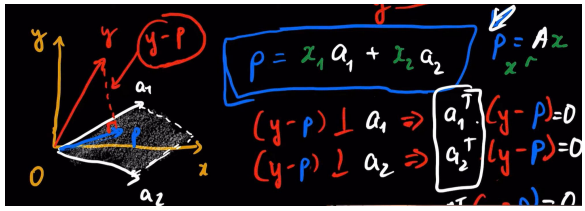


Fig. 1: Behind maths explanation

Final mathematics:

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

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).

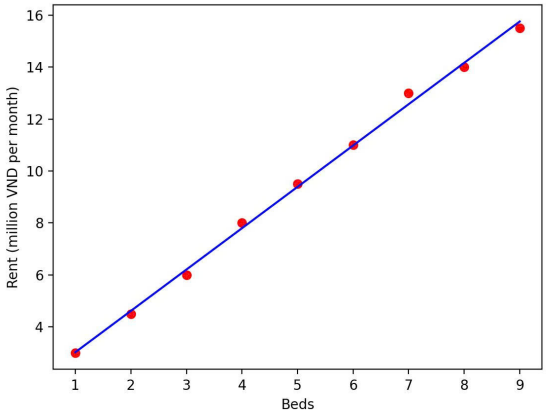


Fig. 2: Example of a 2D fitted line in Linear Regression.

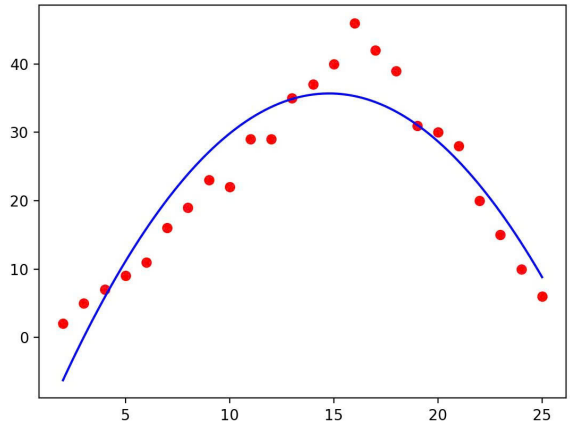


Fig. 3: Example of a 2D fitted parabol in Linear Regression.

C. Applications

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

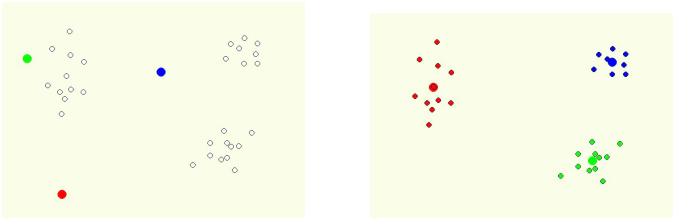
The K-Means algorithm can also take a large number of input features, which means the data can have many dimensions (denoted as D). Each data point $\mathbf{x}_i \in \mathbb{R}^D$ is represented as a vector in a D-dimensional space, and the algorithm aims to find k centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$

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).



(a) (a) Input: $K = 3$ (b) (b) Output

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

B. Mathematical Analysis

The K-Means algorithm takes a set of observations

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

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

1) Choose the number of clusters k.
2) Initialize k centroids randomly.
3) Assign each data point to the nearest centroid.
4) Update each centroid as the mean of the assigned points.
5) Repeat steps 3–4 until centroids no longer change (convergence).

Notation

- N: Number of observations.
- k: Number of clusters.
- x_i: i-th data point.
- μ_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.

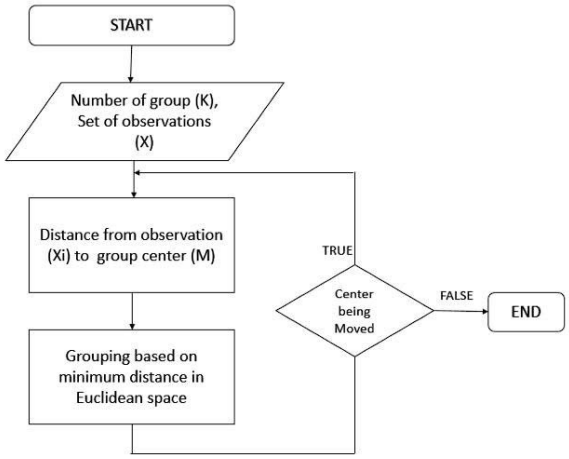


Fig. 5: Flowchart of the K-Means clustering algorithm.

D. Applications

K-Means is widely used in customer segmentation, image compression, anomaly detection, and pattern discovery.

An experiment is conducted where the K-Means algorithm is applied to reduce the size of an image by decreasing the number of unique colors. The algorithm produces a new image that contains fewer color shades compared to the original one.

This experiment is implemented using the Pascal programming language and the SwinGame API. Each pixel in the image contains three elements representing the Red, Green, and Blue (RGB) components. Let each pixel be treated as an observation X, and the total number of pixels in the image corresponds to the total number of observations. Each observation thus has three features — the RGB values.

In this case, the K-Means algorithm is applied to identify K main colors in the image. Each pixel is then reassigned to the nearest color centroid, producing a compressed version of the original image with reduced color diversity.

(a) Original Image (b) $K = 3$ colors

(c) $K = 9$ colors (d) $K = 16$ colors

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

III. Gradient Descent

A. Introduction

Gradient Descent Algorithm

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

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 α is the learning rate.

The first 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.

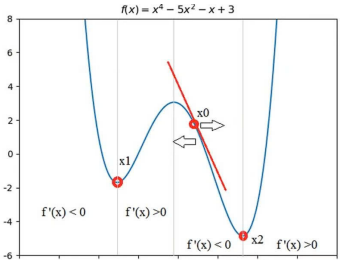
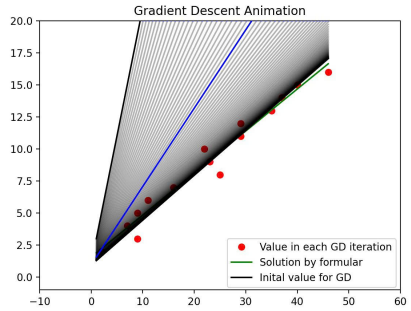
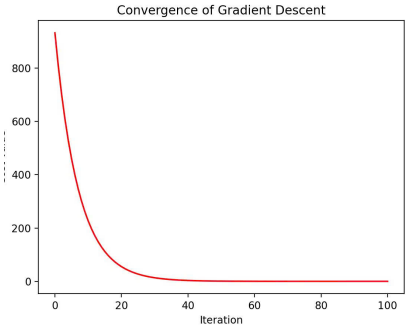


Fig. 7: Example of gradient descent algorithm.



(a) Gradient Descent Iterations



(b) Cost Function Convergence

Fig. 8: Visualization of Gradient Descent. (a) shows the iterative line updates, (b) shows the cost function convergence.

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 $J(\boldsymbol{\theta})$ 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.

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.

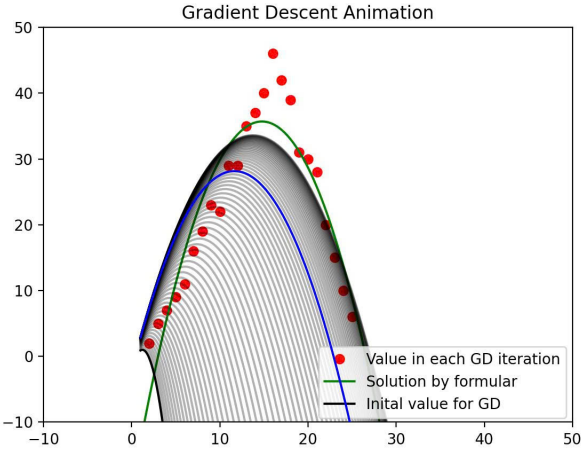


Fig. 9: Visualization of Gradient Descent on a parabolic cost function (1D). The same principle extends to higher-dimensional cost surfaces.

B. Mathematical Analysis

Given a dataset:

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

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

For a new point x_{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.

Fig. 10: KNN classification on the Iris dataset (2D projection using PCA). Different colors represent flower species, and decision boundaries are learned from the training set.

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 28×28 grayscale grid, which is flattened into a 784-dimensional feature vector.

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.

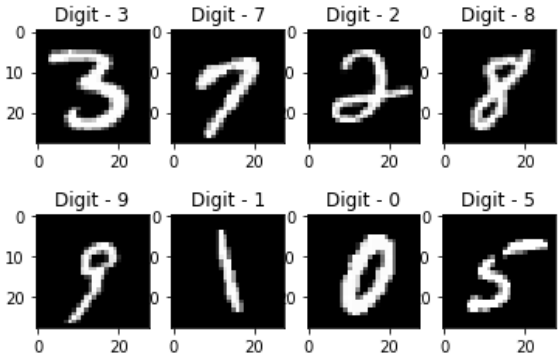


Fig. 11: Visualization of KNN applied to handwritten digit recognition. Points represent digits embedded into 2D space using PCA or t-SNE for visualization.

Advantages:

- Simple and easy to implement.
- No training phase — the model learns directly from the data.
- Effective for small datasets with clear distance-based separation.

Limitations:

- Computationally expensive for large datasets.
- Sensitive to irrelevant or differently scaled features.
- Performance depends heavily on the choice of k and distance metric.