KNN & K-Means Algorithm

Yunsu Han
yunsuhan00107@gmail.com
Bugil Academy

1. Introduction

Machine Learning is one of the artificial intelligence (AI) technologies that have greatly improved the original artificial neural network of the 1950s and has made a significant contribution to several disciplines since the 2010s. In this paper, we look at two simple machine learning algorithms called K-Nearest Neighbors (KNN) and K-Means.

2. Supervised & Unsupervised Learning

a) Methodology

AI programs can execute difficult and outstanding tasks better than human beings, but a program cannot do anything unless it does not have any data fed. To create an AI system, the programmer must feed some data into it. This procedure of feeding the data is called “learning.” The system learns from the data that the programmer gives to it. There are two ways of learning: supervised learning and unsupervised learning.

b) Supervised Learning

Supervised Learning is a way of an AI system learns. When a programmer feeds data to a program using supervised learning, the programmer feeds both the input data and the output data. The goal of supervised learning is to find the pattern, or the function, between the input and the output. Eventually, when a new input data set is given, the system can predict the output data. Programmers or data scientists use supervised learning for programs that they can provide both input and output data, such as export systems in image recognition, forecasting, financial analysis, etc.

c) Unsupervised Learning

Unsupervised learning is another way of an AI system learns. While supervised learning feeds both the input and output data, unsupervised learning feeds only the input data. Its goal is to model the hidden patterns or underlying structure in the given input data to learn about the data. In this case, the programmer only gives the input data because all they have is the input data. Programmers use

unsupervised learning, for instance, pre-processing the data during exploratory analysis or pre-training supervised learning algorithms. Additionally, recommendation systems on Netflix or Spotify use unsupervised learning based on data of a user's history.

3. K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a simple supervised algorithm used for classification. KNN's purpose is to use a database where the data points are separated into several classes to predict the classification of a new point. KNN is also called as a non-parametric, lazy algorithm since it does not take any parameters: it means that KNN does not make any assumptions on the underlying distribution of data points.

a) Principle

KNN works based on feature similarity. It finds where the new point is classified by calculating distances between the new point and all the original data points and finding which one is the closest. When the programmer puts the input data – a positive integer k and a new data point – he or she selects k number of data points, which are closest to the new sample. As a result, the program finds the most common classification of these entries and gives the classification of the new sample. Since the algorithm is simple and straightforward, it is appropriate for machine learning beginners. However, KNN has its downsides, which happen when the value k is an even number. If there are two classes in the database and the value k is, say, four, there is a possibility where the program cannot classify the new sample since the numbers are equivalent.

b) Code Example

The following codes explain how KNN works in detail.

```
In [1]:
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Before we run the code, we must import some necessary packages. Those packages are numpy, pandas (do simple tasks such as creating arrays, sorting tables out, etc.), and matplotlib (draws plots such as scatterplots).

```
In [2]:
url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data"

# Assign column names to the dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'Class']

# Read dataset to pandas dataframe
dataset = pd.read_csv(url, names=names)
```

Next, we need some datasets to perform this algorithm. In this example, we use some data from a website called iris [1].

```
In [3]:
dataset.head()
```

Out[3]:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Figure 1: KNN Data Set Overview

This is a slight overview of how the data set looks.

```
In [4]:
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
```

The next step is to split our dataset into its attributes and labels. The X variable contains the first four columns of the dataset (i.e. attributes) while y contains the labels.

```
In [5]:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

To avoid overfitting, we need to find a good ratio of train dataset: test dataset. This way, the algorithm is tested on unseen data, and so on.

```
In [6]:
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Before making predictions, it is always good to scale the dataset so that we can evaluate every single one of them.

```
In [7]:
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

```
Out[7]:
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

```
In [8]:
y_pred = classifier.predict(X_test)
```

The next step is to train the model and predict in which class the data is included. We set the k value as five and fit the model. The output displays how the parameter is set.

```
In [9]:
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[11  0  0]
 [ 0  8  0]
 [ 0  1 10]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       0.89      1.00      0.94         8
 Iris-virginica       1.00      0.91      0.95        11

       accuracy                           0.97        30
      macro avg       0.96      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

Figure 2: KNN Final Output

The final step is to evaluate the accuracy of the model. We use the confusion matrix to prevent one-sided predictions. This confusion matrix shows not only accuracy but also recall, support, and f1-score.

4. K-Means

K-Means algorithm is a simple unsupervised algorithm used for clustering. Unsupervised learning typically makes inferences using only input vectors from the datasets. Its main purpose is to group similar data points and discover their underlying patterns. To achieve this objective, the K-means algorithm looks for a fixed number, k, of clusters in the dataset. The programmer defines a target number k, which refers to the number of centroids that the program needs in the dataset. A centroid is some sort of location representing the center of the cluster. Namely, the K-means algorithm identifies k number of centroids, and allocates every single data point to the nearest cluster, while keeping the centroids as tiny as possible. The word ‘means’ in K-means refers to averaging of the data, which also means finding the centroid.

b) Code Example

The following code shows an example of the K-means algorithm.

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
%matplotlib inline
```

The following code imports several necessary packages for the K-means algorithm. Pandas and NumPy are packages that most machine learning programs use. Matplotlib allows the programmer to visualize the result in scatterplots. K-means from sklearn package is the most important one in this program; it allows the programmer to use the K-means algorithm itself.

```
In [2]: X= -2 * np.random.rand(100,2)
X1 = 1 + 2 * np.random.rand(50,2)
X[50:100, :] = X1
plt.scatter(X[ : , 0], X[ :, 1], s = 50, c = 'b')
plt.show()
```

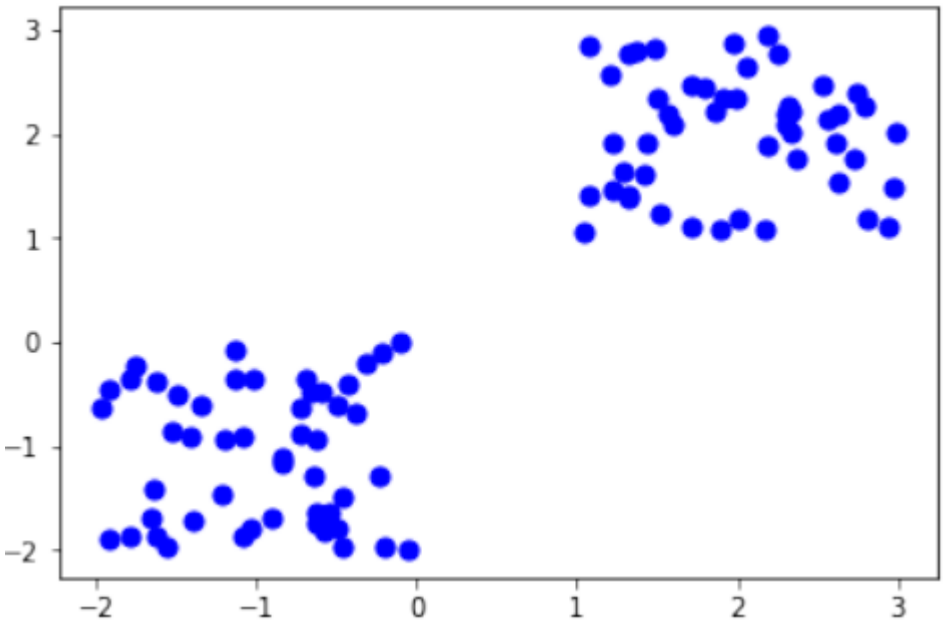

Figure 3: K-Means Graph of Dots

First, create 50 dots each for two different groups. Since the K-means algorithm is unsupervised learning, do not indicate which group is which. Only the input data is used in this algorithm. Visualize the dots by using matplotlib's scatter function.

```
In [3]: from sklearn.cluster import KMeans
Kmean = KMeans(n_clusters=2)
Kmean.fit(X)
```

```
Out[3]: KMeans(algorithm='auto', copy_x=True, init='k-means++',
max_iter=300,
n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
random_state=None, tol=0.0001, verbose=0)
```

The letter "k" in "K-means algorithm" is a number of the clusters. In this program, set the value of k as two. Then, the "fit" function computes k-mean clustering. The output shows how the parameters are set.

```
In [4]: Kmean.cluster_centers_
```

```
Out[4]: array([[-0.97779583, -1.06746949],
[ 1.96953448, 1.99754973]])
```

"cluster_centers_" are coordinates of cluster centers. If the algorithm stops before fully converging, these will not be consistent with labels_.

```
In [5]: plt.scatter(X[ : , 0], X[ : , 1], s =50, c='b')
plt.scatter(-0.94665068, -0.97138368, s=200, c='g', marker='s')
plt.scatter(2.01559419, 2.02597093, s=200, c='r', marker='s')
plt.show()
```

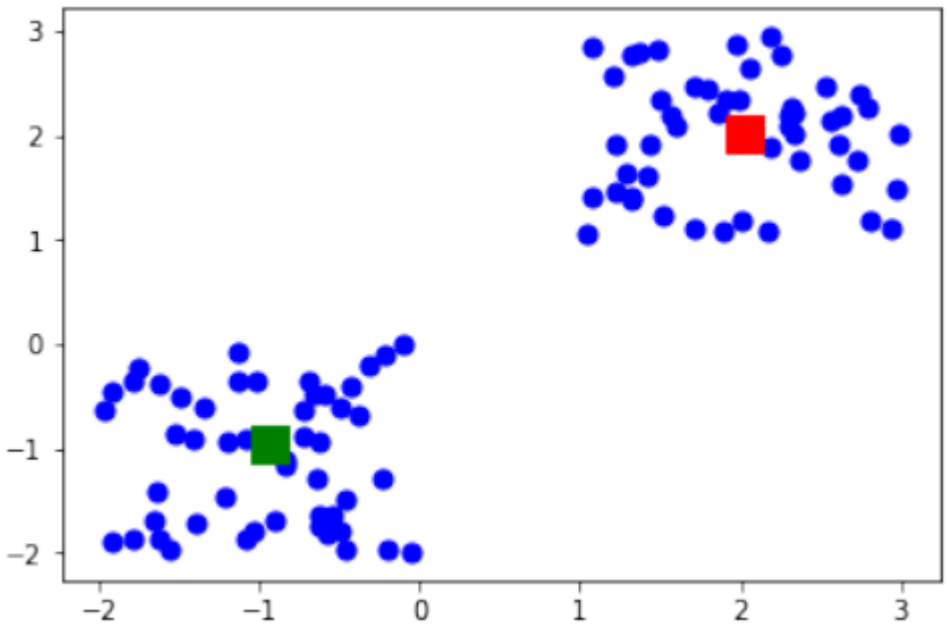

Figure 4: K-Means Graph with New Dots

Lastly, visualize the dots by using two different colors. This is not necessary, but, to show the results to non-experts, a scatterplot is the best example to show the results right away.

```
In [6]: Kmean.labels_
```

```
Out[6]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

"labels_" function labels each point either 0 or 1 since we created two groups beforehand.

```
In [7]: sample_test=np.array([-3.0,-3.0])
second_test=sample_test.reshape(1, -1)
Kmean.predict(second_test)
```

```
Out[7]: array([0])
```

This runs a second test. It tests where the coordinate (-3.0, -3.0) is included in: 0 or 1? It prints out that it is included in group 0, which is correct.

3. Conclusion

In this paper, we looked at the basics of machine learning, supervised and unsupervised learning, and two examples of them: KNN and K-Means.

4. References

[1] “Iris.” Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.