Course: CS550 - Machine Learning and Business Intelligence
Faculty: Prof. Henry Chang

Machine Learning Algorithm: Fall Prediction using KNN

Name: Haripriya A
ID: 19579

NORTHWESTERN POLYTECHNIC
UNIVERSITY

Table of Contents

Introduction

- Machine learning is a method of data analysis investigation that automates logical model structure. Machine learning algorithms are largely used to predict, classify, or cluster.
- Machine Learning
    * Supervised Learning
        - Regression : Linear Regression Algorithm
        - Classification : KNN Algorithm
    * Unsupervised Learning
        - Clustering : K-Mean Algorithm

About K Nearest Neighbour Algorithm

- K-NN is a simple algorithm that can be used when you have a bunch of objects that have been classified or labeled in some way, and other similar objects that haven't gotten classified or labeled yet, and you want a way to automatically label them.
- Procedure of automation
  - Step 1: Define similarity of two objects
  - Step 2: Compare the similarity of a given unrated object with all the classified objects.
  - Step 3: Take the most similar objects and call them neighbors, who each have a "label."
  - Step 4: Determine the "label" of the unrated object.

Calculating K Nearest Neighbour Algorithm Manually

Step 1: Input data

- The data is received from mobile phone which are equipped with sensors like gyroscope and accelerometer.

Accelerometer Data			Gyroscope Data			Fall(+), Not(-)
x1	y1	z1	x2	y2	Z2	+/-
1	2	3	2	1	3	-
2	1	3	3	1	2	-
1	1	2	3	2	2	-
2	2	3	3	2	1	-
6	5	7	5	6	7	+
5	6	6	6	5	7	+
5	6	7	5	7	6	+
7	6	7	6	5	6	+
7	6	5	5	6	7	??

Calculating K Nearest Neighbour Algorithm Manually

Step 2: After we gather K nearest neighbors, we take simple majority of these K-nearest neighbors to be the prediction of the query instance.

We use 8 nearest neighbors and find the K value that is N=8,

$k=\sqrt{N}=\sqrt{8}=3$ that is K=3

Now we calculate the distance between the query instance and all the training samples.

Euclidean distance:

$$d_{tq}^{\ 2} = (X_1^{\ t} - X_1^{\ q})^2 + (X_2^{\ t} - X_2^{\ q})^2$$

Step -3: Calculating K Nearest Neighbour Algorithm using Euclidean distance

x1	y1	z1	x2	y2	Z2	FallOrNot	Accelerometer Sensor + Gyroscope sensor Distance to each neighbor = (Target$_{x1}$-Data$_{x1}$)^2 +(Target$_{y1}$-Data$_{y1}$)^2 +(Target$_{z1}$-Data$_{z1}$)^2 + (Target$_{x2}$-Data$_{x2}$)^2 +(Target$_{y2}$-Data$_{y2}$)^2 +(Target$_{z2}$-Data$_{z2}$)^2 =(7-x1)^2+(6-y1)^2+(5-z1)^2+ (5-x2)^2+(6-y2)^2+(7-z2)^2	Accelerometer Sensor + Gyroscope sensor Distance to each neighbor = Sqrt((Target$_{x1}$-Data$_{x1}$)^2 +(Target$_{y1}$-Data$_{y1}$)^2 +(Target$_{z1}$-Data$_{z1}$)^2 + (Target$_{x2}$-Data$_{x2}$)^2 +(Target$_{y2}$-Data$_{y2}$)^2 +(Target$_{z2}$-Data$_{z2}$)^2) =sqrt((7-x1)^2+(6-y1)^2+(5-z1)^2+ (5-x2)^2+(6-y2)^2+(7-z2)^2)	K =Number of nearest neighbors =sqrt(number of neighbors) =sqrt(number of data samples) =sqrt(9) =3
1	2	3	2	1	3	-	(7-1)^2+(6-2)^2+(5-3)^2+(5-2)^2+(6-1)^2+(7-3)^2=106	10.295630140987	
2	1	3	3	1	2	-	(7-2)^2+(6-1)^2+(5-3)^2+(5-3)^2+(6-1)^2+(7-2)^2=108	10.3923048454133	
1	1	2	3	2	2	-	(7-1)^2+(6-1)^2+(5-2)^2+(5-3)^2+(6-2)^2+(7-2)^2=105	10.7238052947636	
2	2	3	3	2	1	-	(7-2)^2+(6-2)^2+(5-3)^2+(5-3)^2+(6-2)^2+(7-1)^2=101	10.0498756211209	
6	5	7	5	6	7	+	(7-6)^2+(6-5)^2+(5-7)^2+(5-5)^2+(6-6)^2+(7-7)^2=6	2.44948974278318	+
5	6	6	6	5	7	+	(7-5)^2+(6-6)^2+(5-6)^2+5-6)^2+(6-5)^2+(7-7)^2=7	2.64575131106459	+
5	6	7	5	7	6	+	(7-5)^2+(6-6)^2+(5-7)^2+(5-5)^2+(6-7)^2+(7-6)^2=10	3.16227766016838	+
7	6	7	6	5	6	+	(7-7)^2+(6-6)^2+(5-7)^2+(5-6)^2+(6-5)^2+(7-6)^2=7	2.64575131106459	+
7	6	5	5	6	7	??			

Calculating K Nearest Neighbour Algorithm Manually

Step 4: The KNN prediction of the query instance is based on the simple majority of the category of nearest neighbors.

- Our example, We calculated the k value as 3 and calculate the nearest neighbor of the training data and compare the test data.
- The test value is compared with the nearest neighbor and classified as '+' or '-'

Result: In our example the 3 nearest neighbours have '+' and when compared test value classified under '+'

(7-6)^2+(6-5)^2+(5-7)^2+(5-5)^2+(6-6)^2+(7-7)^2=6	2.44948974278318	+
(7-5)^2+(6-6)^2+(5-6)^2+5-6)^2+(6-5)^2+(7-7)^2=7	2.64575131106459	+
(7-5)^2+(6-6)^2+(5-7)^2+(5-5)^2+(6-7)^2+(7-6)^2=10	3.16227766016838	+
(7-7)^2+(6-6)^2+(5-7)^2+(5-6)^2+(6-5)^2+(7-6)^2=7	2.64575131106459	+

Implementing K Nearest Neighbour Algorithm Automation

Step 1: Define similarity between objects

- Our data is classified into two categories ‘+’ and ‘-’ and in program we take the data in the dataset as 1 for ‘+’ and 0 for ‘-’.

```
dataset = [[1,2,3,2,1,3,0],
[2,1,3,3,1,2,0],
[1,1,2,3,2,2,0],
[2,2,3,3,2,1,0],
[6,5,7,5,6,7,1],
[5,6,6,6,5,7,1],
[5,6,7,5,7,6,1],
[7,6,7,6,5,6,1],
[7,6,5,5,6,7,1]]
```

+ Code + Text

Implementing K Nearest Neighbour Algorithm Automation

Step 2: Compare the similarity of a given unrated object with all the classified objects by calculating the distance using Euclidean distance

- Our data is classified into two categories '+' and '-' and in program we take the data in the dataset as 1 for '+' and 0 for '-'.

```
prediction = predict_classification(dataset, dataset[-1], 3)
```

```
[6] def predict_classification(train, test_row, num_neighbors):
        neighbors = get_neighbors(train, test_row, num_neighbors)
        output_values = [row[-1] for row in neighbors]
        prediction = max(set(output_values), key=output_values.count)
        return prediction
```

Implementing K Nearest Neighbour Algorithm Automation

Step 2: Compare the similarity of a given unrated object with all the classified objects by calculating the distance using Euclidean distance

```
[4] def euclidean_distance(row1, row2):
      distance = 0.0
      for i in range(len(row1)-1):
        distance += (row1[i] - row2[i])**2
      print(sqrt(distance))
      return sqrt(distance)
```

Result:

```
10.295630140987
10.392304845413264
10.723805294763608
10.04987562112089
2.449489742783178
2.6457513110645907
3.1622776601683795
2.6457513110645907
```

Implementing K Nearest Neighbour Algorithm Automation

Step 3: Take the most similar objects and call them neighbors, who each have a “label.”

```
[5] def get_neighbors(train, test_row, num_neighbors):
      distances = list()
      for train_row in train:
        dist = euclidean_distance(test_row, train_row)
        distances.append((train_row, dist))
      distances.sort(key=lambda tup: tup[1])
      neighbors = list()
      for i in range(num_neighbors):
        neighbors.append(distances[i][0])
      print(neighbors)
      return neighbors
```

Result:

```
[[7, 6, 5, 5, 6, 7, 1], [6, 5, 7, 5, 6, 7, 1], [5, 6, 6, 6, 5, 7, 1]]
```

Implementing K Nearest Neighbour Algorithm Automation

Step 3: Take the most similar objects and call them neighbors, who each have a “label.”

```
[5] def get_neighbors(train, test_row, num_neighbors):
      distances = list()
      for train_row in train:
        dist = euclidean_distance(test_row, train_row)
        distances.append((train_row, dist))
      distances.sort(key=lambda tup: tup[1])
      neighbors = list()
      for i in range(num_neighbors):
        neighbors.append(distances[i][0])
      print(neighbors)
      return neighbors
```

Result:

```
[[7, 6, 5, 5, 6, 7, 1], [6, 5, 7, 5, 6, 7, 1], [5, 6, 6, 6, 5, 7, 1]]
```

Implementing K Nearest Neighbour Algorithm Automation

Step 4: Determine the "label" of the unrated object.

```
print('Expected %d, Got %d.' % (dataset[-1][-1], prediction))
```

```
Expected 1, Got 1.
```

Result: The test data is classified under 1 which is equal to '+'

Comparison

Result of Manual Calculation: The 3 nearest neighbours have ‘+’ and when compared test value classified under ‘+’

Result using Python Program: The test data is classified under 1 which is equal to ‘+’

By Comparing the result from the Python program and the result of manual calculation, In both the cases we got the same result as ‘+’ for the test data.

Conclusion

- The KNN algorithm is easy to implement. There are only two parameters required to implement KNN i.e. the value of K and the distance function.
- The KNN algorithm doesn't work well with high dimensional data as finding distance between dimensions with categorical features is difficult.
- It is easy to calculate the data manually when there is less data to classify but it will be difficult for calculating the distance and manually classifying large datasets whereas when it comes to python program the large datasets can also be calculated with in less time to predict and classify the data.
- Manual prediction and classification takes more time compared to python program either for small or large datasets.

References

- https://npu85.npu.edu/~henry/npu/classes/data_science/algorithm/slide/knn_from_scratch.html
- https://npu85.npu.edu/~henry/npu/classes/data_science/algorithm/slide/k_nn_example.html
- https://npu85.npu.edu/~henry/npu/classes/data_science/algorithm/slide/knn_from_scratch_explain.html

Google slides URL :

https://docs.google.com/presentation/d/1Ck4MyUiYRnm-r1O8v7z4D2WtnUFsVMv8q9_gg6sVGXI/edit?usp=sharing