Threshold-based Nearest Neighbors: A novel variant of the KNN algorithm

Avni Garg and Alex Gu

January 8, 2024

Dr. Selma Yilmaz

Quarter 2 Project

1 Abstract

The k-nearest neighbors (KNN) algorithm is commonly used machine learning that classifies data based on proximity of neighbors; however, it is unable to grasp more complicated patterns due to its simplicity and the rigidity of the hyperparameter k. To attempt to combat these shortcomings, we created an algorithm called Threshold Nearest Neighbors (TNN). Similar to KNN it considers proximity of points; however, it stops after a threshold t has been reached by summing the reciprocals of the distances rather than looking at a fixed k nearest neighbors. We implemented Euclidean TNN and compared its performance against Manhattan, Euclidean, and Hassanat KNN as well as Euclidean RNN. We tested the algorithms on a dataset we created with sklearn's make_blobs method that had 3 classes, 1550 instance, and 2 attributes. Euclidean TNN revealed itself to be a viable algorithm, with an accuracy of 87.10%, second to Hassanat KNN with 87.42% and tied with Euclidean KNN.

2 Introduction

While algorithms such as k-nearest neighbors are often simple and easy to interpret, they are often inaccurate due to their simplicity. For example, k- nearest neighbors, a generic classification algorithm, is disadvantaged due to its strict locality in the dataset. We believe that the main problem is due to the simple stopping condition. Stopping after checking the k closest instances often leads to inaccurate classification of outliers and of instances on borders. Additionally, skewed datasets with clusters of different densities and size could make an optimal k value hard to find.

To solve this problem, we decided to create an algorithm similar to k-nearest neighbors, but instead of stopping after checking the k-nearest neighbors, it stops after reaching a certain threshold based on how far away the point is. The farther the points are, the more points are needed to make a classification. This is based on the assumption that closer points are better at predicting the class value than points farther away in which case a greater sample size would be helpful.

Similar to k-nearest neighbors, we compute the distance from the instance we want to classify to all other train instances. We then begin considering the train instances from closest to farthest. However, we keep a running sum which we increment by 1/distance for each instance. We stop including these train instances once we either hit a threshold that can be passed as a hyperparameter or when all train instances are already included. Afterwards, we classify the instance by majority vote (unweighted). For the remainder of this paper, we will call this threshold nearest neighbors.

Testing using sklearn's make_blobs functions allows us to create a dataset that can be easily visualized so we can better compare the various algorithms.

3 Related Works

There exist many variations on KNN. Frequently used variations include considering the points in a fixed radius [2, 5] and weighting the neighbors by distance [3]. Other variations include locally adaptive KNNs (LA-KNN) which modifies the optimal k value for each test point and an ensemble approach to KNNs to avoid having to find an optimal k value [6]. Additionally, the distance metric is often changed depending on the model or dataset. Some commonly used distance metrics include Manhattan and Euclidean [1].

It has been shown that out of the KNN variants, locally adaptive KNNs and KNNs with modified distance metrics perform the best [6]. This shows that more accurate models can be achieved by modifying the k value depending on the test points, something our algorithm implements. Additionally, Hassanat KNN had the highest accuracy out of the tested distance metrics for KNNs [1].

Additionally, other studies [4] have attempted to use threshold systems to solve problems related to K-Nearest Neighbors which gives the idea more promise, although no one has tried to create an algorithm similar to ours from scratch.

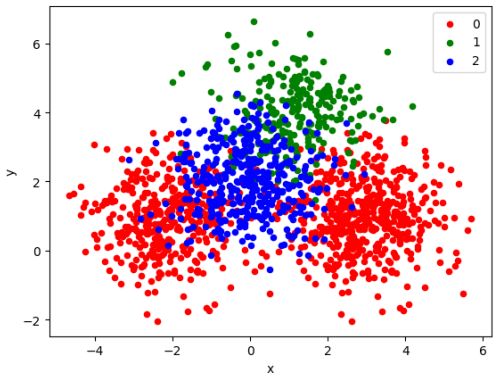


Figure 1: The dataset created by make_blobs. Composed of 3 classes and 1550 instances over 2 attributes. Red represents class 0, green represents class 1, and blue represents class 2.

4 Dataset and Features

For the purposes of performing a preliminary analysis of our algorithm when compared to existing standards we decided to create an artificial dataset to better visualize our results. The data set was motivated by the idea to test our algorithm on a non-uniform dataset since KNNs already perform very well on uniformly distributed datasets. The dataset was created using scikit-learns and consists of 1550 total instances among three classes: 900 instances were of class 0 and split into two separate clusters; 250 were of class 1 and made one cluster; 400 were of class 2 and made a different cluster. All data points had two attributes shown on the x and y axis and had values between -8 and 8. Since the data values were artificially created by us, there was no need for prepossessing of the data. A visualization of our dataset is shown in Figure 1.

We split our data set with a random stratified sample to achieve an 80-20 train-test split. This resulted in there being 1240 train instances and 310 test instances. The split remained constant for each test as to avoid impacting the accuracy due to random chance.

5 Methods

The algorithms we tested were Euclidean KNN, Manhattan KNN, Hassanat KNN, Euclidean fixed radius nearest neighbors (which we will refer to as RNN), and our algorithm, threshold nearest neighbors (TNN).

The k-nearest neighbors is a classification algorithm which classifies instances based on the following: KNN records the distance from that instance to all other train instances utilizing some distance metric. KNN then looks at the k train instances which have the least distance from the instance and classifies it based on the most common class label among those k neighbors.

While KNN allows for a lot of freedom in the sense that there isn't a fixed distance metric, there are still some requirements for a function to be considered a distance metric.

Firstly, it must satisfy nonnegativity. In other words, the distance between some x and y must always be greater than or equal to 0.

$$d(x, y) \geq 0$$

Secondly, it must satisfy the identity of indiscernibles. The distance between x and y must be 0 if and only if $x = y$.

$$d(x, y) = 0 \text{ iff } x = y$$

Thirdly, the distance metric must be symmetric.

$$d(x, y) = d(y, x)$$

Lastly, the distance metric must satisfy the so-called triangle inequality. The distance between x and y summed with the distance between y and z must be greater than or equal to the distance between x and z.

$$d(x, y) + d(y, z) \geq d(x, z)$$

Euclidean KNN, Manhattan KNN, and Hassanat KNN are all implementations of the KNN algorithm but with different distance metrics. The Euclidean distance metric is defined as the following:

$$ED(x, y) = \sqrt{\sum_{i=1}^{n} x_i^2 - y_i^2}$$

The Manhattan distance metric is

$$MD(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

While the previous two distance metrics were types of Minkowski distances in the form of $d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$, Hassanat distance does not take this form. Instead, Hassanat distance is defined as

$$HD(x, y) = \sum_{i=1}^{n} D(x_i, y_i)$$

with $D(x_i, y_i)$ being defined as

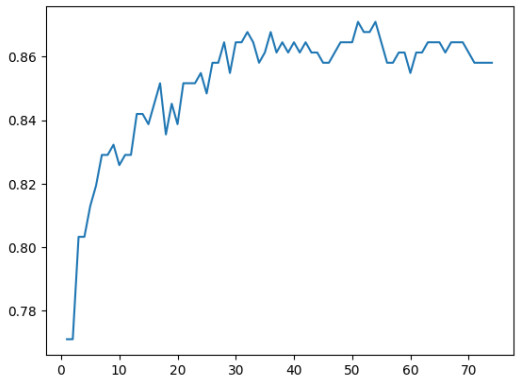
$$D(x_i, y_i) = \begin{cases} 1 - \frac{1+\min(x_i,y_i)}{1+\max(x_i,y_i)} & \min(x_i, y_i) \geq 0 \\ 1 - \frac{1+\min(x_i,y_i)+|\min(x_i,y_i)|}{1+\max(x_i,y_i)+|\min(x_i,y_i)|} & \min(x_i, y_i) < 0 \end{cases}$$

Radius Neighbors is an algorithm similar to KNN but instead of classifying based on the K nearest neighbors, it classifies based on all of the train instances within a fixed radius and picking the most common class [5].

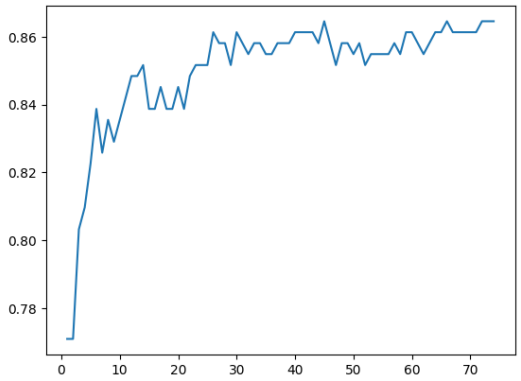
As we described above, threshold nearest neighbors uses somewhat similar logic to KNN, but instead of a fixed K value to decide when to stop, it stops when a threshold is reached. For the n closest instances, if the sum of the reciprocals of the distances of those instances to the reaches a certain threshold, Threshold Nearest Neighbors will stop looking at other train instances and classify based on the most common class. If the threshold is never reached, Threshold Nearest Neighbors will classify based on the most common class out of all of the train instances.

6 Results/Discussion

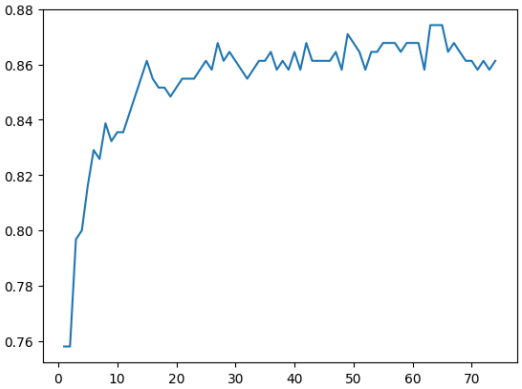
For each of the five algorithms we analyzed, we coded functions to iterate over different hyperparameters to find the best ones for the given dataset. For the KNN algorithms, we wanted to examine enough k values such that a clear downward trend started occuring. Due to this, we iterated over the optimal value for k from 1 to 75.



(a) Euclidean Distance, $k = 56$



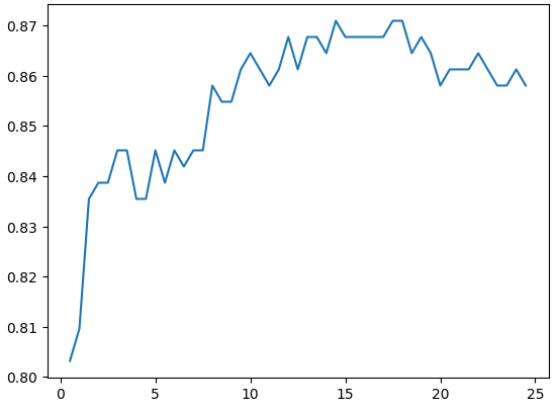
(b) Manhattan Distance, $k = 46$



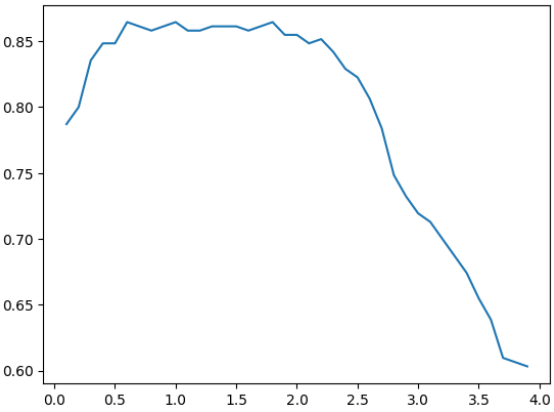
(c) Hassanat Distance, $k = 63$

Figure 2: Accuracy vs k for KNN with different distance metrics.

As shown in Figure 2, when using the Euclidean distance metric, the optimal k value was 56. The optimal k value for the Manhattan distance metric was 46, and the optimal k value of the Hassanat distance metric was 63.



(a) Accuracy vs t for KNN utilizing the Euclidean distance metric. The optimal t value was 14.



(b) Accuracy vs r for KNN utilizing the Euclidean distance metric. The optimal r value was 1.3.

Figure 3: Optimal hyperparameters for non-KNN algorithms.

In a similar fashion, we coded functions to iterate over different values for the threshold t used for Threshold Nearest Neighbors and radius r for Radius Nearest Neighbors. As shown in Figure 3, the optimal t and r values were 14 and 1.3 respectively.

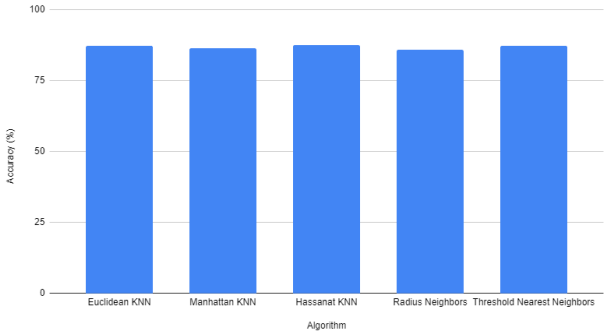


Figure 4: Accuracy vs Algorithm. Hassanat performed the best with Euclidean KNN and Threshold Nearest Neighbors tieing for second.

Due to our dataset having three different classes, our primary performance metric was accuracy. As depicted in Figure 4, when testing these algorithms on the dataset created by make_blobs, we obtained the following accuracies: Euclidean KNN had an accuracy of 87.10%, Manhattan KNN had an accuracy of 86.45%, Hassanat KNN had an accuracy of 87.42%, Radius Neighbors utilizing Euclidean distance had an accuracy of 85.81%, and Threshold Nearest Neighbors utilizing Euclidean distance had an accuracy of 87.10%.

The algorithms also had the following confusion matrices:

KNN EUCLIDEAN

Accuracy: 0.8709677419354839

	Predicted Class 0	Predicted Class 1	Predicted Class 2
Actual Class 0	167	1	12
Actual Class 1	1	41	6
Actual Class 2	12	8	62

KNN MANHATTAN
Accuracy: 0.864516129032258

	Predicted Class 0	Predicted Class 1	Predicted Class 2
Actual Class 0	167	1	13
Actual Class 1	1	40	6
Actual Class 2	12	9	61

KNN HASSANAT
Accuracy: 0.8741935483870967

	Predicted Class 0	Predicted Class 1	Predicted Class 2
Actual Class 0	169	1	10
Actual Class 1	1	39	7
Actual Class 2	10	10	63

FIXED RADIUS EUCLIDEAN
Accuracy: 0.8580645161290322

	Predicted Class 0	Predicted Class 1	Predicted Class 2
Actual Class 0	167	1	13
Actual Class 1	1	39	7
Actual Class 2	12	10	60

TNN EUCLIDEAN
Accuracy: 0.8709677419354839

	Predicted Class 0	Predicted Class 1	Predicted Class 2
Actual Class 0	167	1	12
Actual Class 1	1	40	5
Actual Class 2	12	9	63

The confusion matrices are all extremely similar, which is within expectations due to the similar nature of all of the algorithms. This is also likely due to the relative simplicity of our dataset due to it being produced by the make_blobs method. As such, we cannot make any clear distinctions between the algorithms by analyzing the confusion matrices.

7 Conclusion/Future Work

On our dataset, Hassanat KNN performed the best with an accuracy of 87.42%. Euclidean KNN and Threshold Nearest Neighbors using the Euclidean distance metric followed close behind with an accuracy of 87.10%. Manhattan KNN came in fourth with an accuracy of 86.45% and Radius Neighbors using Euclidean distance performed the worst with an accuracy of 85.81%.

Hassanat KNN likely performed the best due to its steep dropoff as the distance become shorter, providing a steeper metric. Additionally, the Hassant distance metric is optimized for use in KNNs. Euclidean KNN and Euclidean Threshold Nearest Neighbors may have performed the next best due to the Euclidean distance metric being the Minkowski $p = 2$ distance. This also shows that our algorithm can perform at least as well as the standard KNN algorithm.

Following this train of thought, Manhattan KNN was fourth. This could be due to Manhattan distance being the Minkowski $p = 1$ distance.

However, Radius Nearest Neighbors performing the worst doesn't quite align with this. This may be due to the relative sparseness of our dataset, making the fixed radius less adaptable and hence performing worse.

Future work would be involve testing on more complex and real world datasets on our model. Our preliminary analysis has indicated that threshold nearest neighbors is a viable algorithm for real world applications. Additionally, future work would need to be done on the best distance metric to optimize it.

8 Contributions

A Garg worked primarily on writing the code for the algorithms and testing along with plots and visuals. A Gu did the majority of the writing and analysis. Both authors verified each others work and contributed equally to the initial research.

References

[1] Haneen Arafat Abu Alfeilat, Ahmad B.A. Hassanat, Omar Lasassmeh, Ahmad S. Tarawneh, Mahmoud Bashir Alhasanat, Hamzeh S. Eyal Salman, and V.B. Surya Prasath. Effects of distance measure choice on k-nearest neighbor classifier performance: A review. *Big Data*, 7(4):221–248, December 2019.

[2] Xinye Chen and Stefan Güttel. Fast and exact fixed-radius neighbor search based on sorting. 2023.

[3] Padraig Cunningham and Sarah Jane Delany. k-nearest neighbour classifiers: 2nd edition (with python examples). 2020.

[4] K. Mouratidis, D. Papadias, S. Bakiras, and Yufei Tao. A threshold-based algorithm for continuous monitoring of k nearest neighbors. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1451–1464, November 2005.

[5] Volker Turau. Fixed-radius near neighbors search. *Information Processing Letters*, 39(4):201–203, 1991.

[6] Shahadat Uddin, Ibtisham Haque, Haohui Lu, Mohammad Ali Moni, and Ergun Gide. Comparative performance analysis of k-nearest neighbour (knn) algorithm and its different variants for disease prediction. *Scientific Reports*, 12(1), April 2022.