Optimizing k-nearest neighbor with color detection application

Omer Al Sumeri  
*Electrical and Computer Engineering*   
*California State University, Fresno*Fresno, CA  
omeralsumeri@gmail.com  
  
Luis Villanueva  
*Electrical and Computer Engineering*  
*California State University, Fresno*Fresno, CA  
villanueval@mail.fresnostate.edu  
  
Drake Robert Sheela  
*Electrical and Computer Engineering*  
*California State University, Fresno*Fresno, CA  
drake3861@mail.fresnostate.edu  
  
Hitender Singh Sidhu  
*Electrical and Computer Engineering*  
*California State University, Fresno*Fresno, CA  
hss08@mail.fresnostate.edu

Jacob Luscomb  
*Electrical and Computer Engineering*  
*California State University, Fresno*Fresno, CA

parchment@mail.fresnostate.edu

Shahab Tayeb, Phd  
*Electrical and Computer Engineering*  
*California State University, Fresno*Fresno, CA  
tayeb@mail.fresnostate.edu

*Abstract—*This paper presents an improvement to the efficiency of the k-nearest neighbor algorithm by reducing its time and space complexity while keeping accuracy high. KNN is a popular classification algorithm in machine learning that classifies a new data point by considering the k-number of nearest neighbors surrounding it. However, the naive implementation of k-nearest neighbor requires computing the distance of the entire dataset to the unknown point, resulting in significant time and space consumption. To address this issue, a new implementation of k-nearest neighbor is proposed for datasets that have distinct clusters of condensed datapoints. The new implementation reduces the clusters of condensed datapoints into a single datapoint. The value k is improved from square root of n to 1; where n is the total number of points in the dataset and the 1 comes from the single closest datapoint to the unknown point. This new implementation significantly reduces the time and space complexity of the k-nearest neighbor algorithm. The clusters of data now represented by a single point reduces the number of comparisons and the amount of data the machine needs to store to use the algorithm. A color detection application was used to analyze the results from the naive to the improved implementation. Time complexity, space complexity, and an accuracy experiment were done to test the improved implementation from the naive. It is concluded that the improved implementation outperforms the naive in both space and time complexity while keeping the same accuracy.

*Keywords— k-Nearest Neighbor, Color Detection*

# Introduction

Several types of machine learning algorithms exist, two popular are supervised learning and unsupervised learning. Supervised learning is a way of teaching an algorithm to classify an input with pre-classified data. In unsupervised learning, an algorithm will find patterns in unclassified data without any human interaction. The algorithm, k-nearest neighbor (KNN) is considered a supervised learning algorithm due to it being given previously classified data [1]. KNN is used for classification problems by having pre-classified data points which can be compared to the input data point. Finding a specific amount of closest number of neighboring values will determine what classification will be given to the new input.

The KNN algorithm has three main parts which make up the function of the algorithm. First, the distances from the new input point to all other already preloaded classified points are calculated by using Euclidean distance. The Euclidean distance is a formula which calculates the distance between two points in the cartesian coordinate system [2]. This formula uses the Pythagorean theorem which can be seen in Fig. 1. [3].

Text, letter

Description automatically generated

Fig. 1. Euclidean Distance Formula [3]

These distances are then stored inside of a list and sorted from smallest to largest. With the list now sorted, the closest data points to the input point are at the beginning of the list. Depending on the set k-value, the first k-values in the list will be retrieved. A tally is done on which classification group is closest. Whichever group has more tallies, the new input point is added to that group due to the point being closer to multiple values of that classified group [4].

The k in KNN is a value which is used to select how many neighboring close points to compare the new point to. If a low k-value is selected then the accuracy of the classification may decrease, while having a large k-value may improve accuracy but can increase bias due to training errors. Usually, k is an odd number to prevent two groups having the same number of tallies which would result in a non-classification [2].

KNN is used in the industry for many applications such as predicting weather, stock market predictions, and health predictions. Every algorithm has advantages and disadvantages; with KNN, the algorithm is known for a simple implementation while giving meaningful results. The downfall of the algorithm comes when the given data set is large, leading to longer computation times due to the large amount of data [5].

In this paper, KNN is used to classify color. A wide range of colors can be represented by a combination of three different colors: red, green, and blue (RGB). Together these colors can be combined to make new colors such as yellow or pink. These three colors can be represented on a three-axis graph. The range of each axis would only be limited to values from 0 to 255. By implementing color on a three-axis graph with a range of 0 to 255 will give a possibility of 16,777,216 different colors. By translating a color into values, it is possible to now feed this type of data into an algorithm to help the algorithm quantify what color the computer is reading [6].

# Limatations of Existing method

The limitations that exist for the naive KNN algorithm is the computation required to classify a new data point. The new data point must be compared to every pre-classified data point. A calculation for each data point using the Euclidean distance to determine the distance will take extra time even if it only needs to use k-datapoint distances, depicted in Fig. 2. Reference [7] shows how they used Euclidean distance to calculate distances for each data point of chili leaves using the Hue, Saturation, and brightness value of color. This means that the distance for each new leaf input must be calculated for each data point in the dataset.

A diagram of different colored dots

Description automatically generated

Fig. 2. Illustration of distances that need to be calculated for naive KNN

The k-value that the algorithm will use also increases the number of comparisons that will need to be happen between the dataset to the new datapoint to determine its classification. Using a range of k-values can influence the accuracy of the algorithm. In the KNN Implementation shown in reference [8], it is evident that with the k-value ranging there is also a range of accuracies that can occur. By generalizing the values, k can be reduced to determine the smallest distance and using the smallest distance as the choice of classification.

In a typical naive KNN algorithm the k-value is chosen with the , n being the size of the data points. So, as the size of the data points increase so will the k value. Increasing the time and space complexity for the calculations as the dataset becomes larger.

The time and space complexity for the for naive KNN algorithm is , where n is the total number of data points and d is the total number of features in the dataset [9]. Features, in the case for this paper, means the different available classifications.

# Proposed Method

The proposed method for improving KNN will be representing the clusters of data to a single point. The single representing point for each classification will be the average of all the datapoints in that classification. Every classification in the dataset will have one representing value, the representing values will be used to then to measure the distance to the unknown point. The classification that has the least measured distance is then the classification of the unknown point.

If every classification is represented by a single data point, then the distance that needs to be calculated is only the number of classifications as shown in Fig. 3.

A diagram of a line with different colored lines

Description automatically generated with medium confidence

Fig. . Illustration of distances that need to be calculated for improved KNN

With the proposed method, the choice of k will go from , n being the size of the data points, to 1, the shortest distance calculated from unknown point to the classifications. The proposed method reduces time complexity and space complexity. The time complexity is reduced because the number of distances calculated is reduced. The improved time complexity will be , C being the number of classifications in a dataset. The space complexity is reduced because the algorithm is not required to store the entire dataset, only the representation point for each classification. The improved space complexity will be again.

In the context of color detection, each color will have a single representing point. The single representing RGB value chosen for the specific color will be a general representation of that color. The unknown color will have three values for it to be plotted on the RGB graph. The distance will be taken from the unknown color to all the known labeled colors on the graph. All the distances will be calculated and stored in a list. The main color that has the shortest distance to the unknown color will now be the color for the unknown color.

# Methodology

The following implementation of the naive and proposed KNN algorithms will be tailored for the color detection application. The naive and improved programs will be able to detect 8 colors. The 8 colors are red, green, blue, yellow, green, orange, purple, black, and white.

For the naive and improved implementation, a color point class needs to be created. The attributes of the color point class are shown in Fig. 4. The three integers, r representing red, g representing green, and b representing blue, are the RGB attribute of the individual color point. The distance which is of data type double is used to store the distance from the individual color point to the unknown color point. The color given to the respective input will be held in the color type attribute.

A black text on a white background

Description automatically generated

Fig. . Class and attributes created for a single-Color point

For the naive implementation, color points need to be generated within a specific range. Table 1 contains the RGB value ranges for each color classification.

Table 1. RGB value ranges for each Color Classification in the Naive Implementation

|  |  |  |  |
| --- | --- | --- | --- |
| Color | Red  Min-Max | Green  Min-Max | Blue  Min-Max |
| Red | 225 - 255 | 0 - 30 | 0 - 30 |
| Blue | 0 - 30 | 0 - 30 | 225 - 225 |
| Yellow | 225 - 255 | 225 - 255 | 0 - 30 |
| Green | 0 - 30 | 225 - 255 | 0 - 30 |
| Orange | 225 - 255 | 113 - 143 | 0 - 30 |
| Purple | 112 - 142 | 0 - 30 | 225 - 255 |
| Black | 0 - 30 | 0 - 30 | 0 - 30 |
| White | 225 - 255 | 225 - 255 | 225 - 255 |

The process of the naive algorithm is summarized as follows:

1. Find the distance from all the points in the dataset to the unknown point as shown in Fig. 5
2. Sort all the distances using a sorting algorithm, placing the values in an ascending-ordered list
3. Classify the unknown color as shown in Fig. 6
   1. Take the square root of the size of the data set to find k
   2. Count the occurrence of each color up to k
   3. Find the color with the greatest occurrence
   4. Set the unknown color to the color that occurred the most

A math equations on a white background

Description automatically generated

Fig. . Pseudocode for calculating the distance from unknown color to all colors in the dataset for the Naive implementation

A black and white text on a white background

Description automatically generated

Fig. 6. Pseudocode for classifying unknown color using the naive implementation

For the improved implementation, the representing point needs to be set for each classification. In the case of the color application, a representing RGB value will need to be created for each color. The representing values were chosen by selecting the general representation of each color. Table 2 contains each representing RGB value for each color.

Table 2. Representing RGB value for each color classification

|  |  |  |  |
| --- | --- | --- | --- |
| Color | Red Value | Green Value | Blue  Value |
| Red | 255 | 0 | 0 |
| Blue | 0 | 0 | 255 |
| Yellow | 255 | 255 | 0 |
| Green | 0 | 255 | 0 |
| Orange | 255 | 128 | 0 |
| Purple | 127 | 0 | 255 |
| Black | 0 | 0 | 0 |
| White | 255 | 255 | 255 |

The process of the improved algorithm is summarized as follows:

1. Find the distance from all the color classifications to the unknown point as shown in Fig. 7
2. Classify the unknown color as shown in Fig. 8
   1. Set the minimum distance to the first Color Classification
   2. Search through the whole Color Classification list to find the minimum distance calculated
   3. Set the unknown color the color in the Color Classification list that has the least distance to the unknown point

A black text on a white background

Description automatically generated

Fig. . Pseudocode for calculating the distance from unknown color to all colors in the dataset for the improved implementation

A close-up of a text

Description automatically generated

Fig. . Pseudocode for classifying unknown color using the improved implementation

# Results/Analysis

To test the improved method from the naive method, three experiments were conducted. The experiments tested the space complexity, time complexity, and the accuracy. The space complexity tests the amount of memory or predefined data the algorithm requires to execute. The time complexity tests the number of computations that the algorithm will process. The accuracy test assesses if the improved algorithm maintains accuracy compared to the naive algorithm.

Both programs were implemented in the programming language C++. The testing was performed on a computer system with a macOS Mojave operating system, an Intel® Core™ i7-4980 HQ CPU @ 2.80GHz, 16GB DDR3 1600MHz Micron Technology RAM, and an AMD Radeon R9 M370X 2GB GPU. The development platform used was Code::Blocks 13.12 mac version. For timing purposes, chrono version 8.0.0 was used.

## Time Complexity Experminent

For the time complexity experiment, two versions of the naive KNN algorithm were measured against the improved KNN algorithm. One version was using a nonoptimal sorting algorithm being bubble sort which has a time complexity of . The other version of the naive KNN algorithm had a more optimal comparison-based sorting algorithm, with a time complexity of The purpose of including two different sorting algorithms is to show there will be a huge delta no matter the sorting algorithm used in the naive KNN algorithm.

Fig. 9 is a graph that displays the execution times for the two naive algorithms with the different sorting algorithms implemented and the improved KNN algorithm (Table of times in Appendix A). The input number of unknown colors to be classified starts from 1,000, with an increment of 1,000, all the way up to 10,000. The naive algorithm with bubble sort takes much longer than the naive algorithm with sort and the Improved KNN. The difference can be seen between the naive algorithm with sorting implemented and the improved KNN, but it is hardly visible due to the naive algorithm with bubble sort implemented high execution times.

A graph with a line

Description automatically generated

Fig. . Naive and Improved Algorithms using Two Different Sorts

Fig. 10 is a graph that displays the execution times for the naive algorithm with a sorting algorithm implemented and the improved KNN algorithm (Table of Times in Appendix Table A). The input number of unknown colors to be classified is from 1,000, with an increment of 1,000, all the way to 10,000. The improved KNN is on average 300 times faster than the naive KNN algorithm with the most optimal comparison-based sorting algorithm. So much so, the growth of improved KNN is not visible due to delta between the two algorithms. The overall resultant time complexity of the improved algorithm is independent of the number of data points. This resulted in a constant O(C) time complexity, with C being the number of predefined color classifications.

A graph with a blue line

Description automatically generated

Fig. .Naive and Improved Algorithms using Sort

## Space Complexity Experiment

The original naive algorithm needs to be fed preclassified data points to make a classification, the more preclassified points given to the algorithm improves accuracy but comes at the cost of taking up more memory. The naive algorithm takes O(n \* C), where n is the number of preclassified points each group of colors will be populated with and C comes from there being a constant amount of specified colors. There are eight classifications for colors, each of which require a list of preclassified color points making the space complexity O(8\*N). When classifying 1000 points with 100 preclassified points for each color category equates to 800 preclassified data points which takes 1.8 MB as seen in Fig. 11.

A picture containing text, multimedia software, software, screenshot

Description automatically generated

Fig. 11. Naive Algorithm Classifying 1000 Points with 8 Classification Colors with 100 Pre-Classified Points Each

Increasing the naive algorithm’s accuracy can be done by also increasing the number of pre-classified data points. By providing a large amount of preclassified data points the algorithm has a larger dataset to compare itself to, having a better chance of finding a close match. This also increases the memory rapidly. Classifying 1000 new colors with 1 million preclassified data points per color results in 8 million preclassified data points which takes 813 MB as seen in Fig. 12.

A screenshot of a computer

Description automatically generated with low confidence

Fig. 12. "Classification of 1000 new data points, using 1 million pre-classified data points per color classification."

The improved algorithm only stores a list of eight predefined color points making the space complexity O(8). The value will always be constant depending on how many colors the algorithm needs to detect which means the memory needed to run will always be very small and constant, by doing so the memory required to classify 1000 points with eight classification points only 1.6 MB are needed which is less than the naive algorithm MB as seen in Fig. 13. This value will stay constant no matter the amount of new data points.

A screen shot of a graph

Description automatically generated with low confidence

Fig. . Improved algorithm classifying 1000 points with 8 classification color points

## Accuracy Experiment

For the Accuracy experiment, the naive and improved algorithm were tested with the same RGB values. For each color, five different variations of that color were tested. Test 1 is the lightest of the specific color and the color progressively gets darker until Test 5. The color of the tested RGB value is filled into the background of the cell (Table of tested values in Appendix Table B & C). Beside it, is the color the algorithm detected. The total number of tests done on each algorithm is 40.

Fig. 14 and Fig. 15 contain the results for the accuracy experiment using the naive algorithm. The RGB values that were detected incorrectly are (255,102,102), (102,102,255), and (153,153,0). The RGB value (255,102,102) is light red but the naive algorithm detected it as Orange. The RGB value (102,102, 255) is light blue but the naive algorithm detected it as Purple. The RGB value (255,102,102) is dark yellow but the naive algorithm detected it as Orange. 37/40 colors were detected correctly, concluding a 92.5% accuracy rate for the naive algorithm.

A chart of different colors

Description automatically generated

Fig. 14. Accuracy test outcomes for Red, Green, Blue and Purple using naive implementation.

A chart of different colors

Description automatically generated

Fig. 15. Accuracy test outcomes for Yellow, Orange, Black, and White using the naive implementation.

Fig. 16 and Fig. 17 contain the results for accuracy experiment using the improved algorithm. The RGB values that were detected incorrectly are (255,102,102), (102,102,255), and (153,153,0). All colors that were detected incorrectly by the naive algorithm were also detected incorrectly by the improved algorithm. 37/40 colors were detected correctly, concluding a 92.5% accuracy rate for the naive algorithm.

A chart of different colors

Description automatically generated

Fig. 16. Accuracy test outcomes for Red, Green, Blue, and Purple using the improved implementation.

A chart of different colors

Description automatically generated

Fig. 17. Accuracy test outcomes for Yellow, Orange, Black, and White using the improved implementation.

The naive algorithm has a 92.5% accuracy rate and the improved algorithm has a 92.5% accuracy rate. According to the experiment performed by testing five different variations of the eight colors that are being classified, there is no accuracy difference between the naive and improved algorithms.

# Conclusion

KNN is an algorithm that classifies a new data point by calculating the distance from the new data point to all the existing data points in the dataset. The k-number closest distances are then used to classify the data point. The standard method for calculating k is n, n being the total number of data points in the dataset. The improved KNN algorithm optimizes the k value to be 1 in datasets that have distinct clusters of condensed data points. Each of those clusters can be represented by a single point, reducing the time and space complexity consumption of the algorithm.

To test the improved algorithm, the application of classifying color was chosen. Three experiments were conducted, space, time, and accuracy to show that the improved algorithm outperforms the naive implementation of KNN. The improved algorithm has a final time complexity of O(C), C being the number of color classifications we have in the program. The number of color classifications is 8, so the final time complexity is O(8). The space complexity is also O(C), since only the single-color classification data point needs to be stored. Again, the space complexity in this case is O(8). The accuracy test demonstrated that the same level of accuracy is kept between the naïve and the improved implementation all while making significant development upon the space and time complexity.

The experiments hence shows, that the improved algorithm outperforms the naive algorithm in classifying new data points on datasets that have a structure of condensed data groups.

##### Appendix

1. Table 3. Execution of times of the Naive and Improved KNN algorithms

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Unknown Colors Classified | Naive KNN using Bubble Sort Execution Time (seconds) | Naive KNN using Sort Execution Time (seconds) | Improved KNN Execution time  (seconds) |
| 1,000 | 6.84933 | 0.195698 | 0.000652 |
| 2,000 | 13.7052 | 0.398876 | 0.001299 |
| 3,000 | 20.9106 | 0.596872 | 0.001926 |
| 4,000 | 27.7479 | 0.788861 | 0.002422 |
| 5,000 | 34.5055 | 0.985374 | 0.003288 |
| 6,000 | 41.5026 | 1.19979 | 0.004632 |
| 7,000 | 48.7438 | 1.39622 | 0.002874 |
| 8,000 | 54.5085 | 1.58251 | 0.005347 |
| 9,000 | 61.5478 | 1.77582 | 0.005891 |
| 10,000 | 68.5979 | 1.97525 | 0.007138 |

1. Table 4. RGB values that were tested for Red, Green, Blue, and Purple in the Accuracy experiment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tested  Color | Red | Green | Blue | Purple |
| Test 1 | 255,102,102 | 102,255,102 | 102,102,255 | 178,102,255 |
| Test 2 | 255,51,51 | 51,255,51 | 51,51,255 | 153,51,255 |
| Test 3 | 255,0,0 | 0,255,0 | 0,0,255 | 127,0,255 |
| Test 4 | 204,0,0 | 0,204,0 | 0,0,204 | 102,0,204 |
| Test 5 | 153,0,0 | 0,153,0 | 0,0,153 | 76,0,153 |

1. Table 5. RGB values that were tested for Yellow, Orange, Black, and White in the Accuracy experiment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tested  Color | Yellow | Orange | Black | White |
|  |  |  |  |  |
| Test 1 | 255,255,102 | 255,178,102 | 51,25,0 | 226,223,235 |
| Test 2 | 255,255,51 | 255,153,51 | 0,51,51 | 226,217,218 |
| Test 3 | 255,255,0 | 255,128,0 | 25,0,51 | 255,252,218 |
| Test 4 | 204,204,0 | 204,102,0 | 51,0,51 | 224,224,224 |
| Test 5 | 153,153,0 | 153,76,0 | 0,0,0 | 255,255,255 |

##### References

1. Alteryx, “Supervised vs. unsupervised learning; which is best?,” Alteryx, 2023. [Online]. Available: https://www.alteryx.com/glossary/supervised-vs-unsupervised-learning. [Accessed: 26-Mar-2023].J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
2. O. Harrison, “Machine learning basics with the K-nearest neighbors algorithm,”Medium,14-Jul-2019.[Online].Available:https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. [Accessed: 26-Mar-2023].
3. M. Mancini, “What is the formula for distance?,” HowStuffWorks Science, 08-Mar-2023. [Online]. Available: https://science.howstuffworks.com/math-concepts/distance-formula.htm. [Accessed: 26-Mar-2023].
4. GeekforGeeks, “K-nearest neighbours,” GeeksforGeeks, 14-Mar-2023. [Online]. Available: https://www.geeksforgeeks.org/k-nearest-neighbours/. [Accessed: 26-Mar-2023].
5. ijera editorF. publisher, “DI35605610,” Issuu, 28-Sep-2013. [Online]. Available:https://issuu.com/www.ijera.com/docs/di35605610. [Accessed: 26-Mar-2023].
6. Texas Wesleyan University, “Libguides: Images: Color codes,” Color Codes - Images - LibGuides at Texas Wesleyan University, 01-Jun-2022. [Online].Available:https://txwes.libguides.com/c.php?g=978475&p=7075536. [Accessed: 26-Mar-2023].
7. A. Qur’ania, P. Harsani, Triastinurmiatiningsih, and D. P. Por, “Hue Saturation Value (HSV) Color Detection to Identify Nutrient Deficiencies in Chili (Capsiumannuum L) using K-Nearest Neighbor,” Journal of Advanced Research in Dynamic and Control Systems, vol. 12, pp. 103–108, Jan. 2020, doi: 10.5373/jardcs/v12sp6/sp20201012.
8. A. R. Isnain, J. Supriyanto, and M. Kharisma, “Implementation of K-Nearest Neighbor (K-NN) Algorithm For Public Sentiment Analysis of Online Learning,” DOAJ (DOAJ: Directory of Open Access Journals), Apr. 2021, doi: 10.22146/ijccs.65176
9. R.Shah. “Introduction to k-Nearest Neighbors (kNN) Algorithm. Meduim. https://ai.plainenglish.io/introduction-to-k-nearest-neighbors-knn-algorithm-e8617a448fa8#:~:text=Space%20Complexity%3A%20The%20space%20complexity,of%20features%20in%20the%20dataset. [Accessed Dec. 21, 2023]