K-Closest Clusters (KCC): A Novel Cluster-Based Alternative to KNN

Julia Chen and Yuan Yin

Thomas Jefferson High School for Science and Technology

Machine Learning 1

Dr. Yilmaz

January 26, 2025

Abstract

We wanted to lessen the limitations of popular machine learning models that classify quantitative data: K-Nearest Neighbors and K-Means in particular. In order to do so, our approach included combining aspects of the two by taking the clustering aspect of K-Means and the classification based on the nearest neighbors aspect of KNN. Our methodology was composed of creating a k number of clusters within a dataset and appointing the centroid of each cluster with the majority class label of that cluster. Then, to classify, we implemented KNN with k=1, considering each centroid as a neighbor. This algorithm is known as K-Closest Clusters (KCC). The results show that KCC achieved a slightly better accuracy, precision and recall than the existing KNN algorithm and that KCC is significantly more efficient at classifying test instances than KNN. This indicates that KCC is more practical than KNN as a classifier, especially when used for large datasets.

I. Introduction

There are limitations to both the K-Nearest Neighbors algorithm and the K-Means algorithm. The two biggest problems in using KNN as a classification algorithm are issues with time complexity and memory.

Because KNN requires calculations of the distance between the instance being classified and every other existing data point, classification would require a lot of time in cases where the dataset is very large and/or when many instances need to be classified. Additionally, KNN stores all these distance calculations in fast memory as opposed to simply storing a model. As a result, users may run into issues such as hitting the memory limit.

On the other hand, K-Means is limited to linear cluster boundaries and has a hard time classifying instances in a dataset with weird geometry and/or unbalanced class proportions. This limits K-Means' potential to be used as a classification model for a wide variety of datasets. Because K-Means works by clustering data points in even clusters, it is unable to account for strange shapes in the data distribution.

The input to our algorithm is a dataset of instances with two attributes. The dataset was randomly regenerated using make_blobs. We then use an approach that combines KNN and K-Means to output a predicted class value.

II. Related Work

In a study, Deng et al. [1] develops a novel kNN algorithm that can handle large datasets. kNN is computationally expensive, especially for large datasets, as the distance between the given test instance and each instance in the dataset must be calculated to classify the test instance. To reduce the amount of computation necessary for using the kNN algorithm on large datasets, Deng et al. first uses clustering and subsequently runs kNN on the relevant cluster to classify the instance.

[2] discusses a methodology to mitigate the issue of clustering when the clusters within the data are irregular and/or widely spaced apart. They do so by finding the nearest neighbor of

each data point and finding core points by analyzing the strength of each neighboring pair. Then, clusters are built around these core points. While a strength of this method would be the ability to cluster points that are not densely compacted together, a weakness would be that the process is time consuming. In comparison to our model, it may be able to cluster more accurately, but take longer to do so as a result.

In a study, Alizadeh [3] uses a combined approach of K-Means and KNN to classify instances. Their research is similar to ours in that they use K-Means to create clusters and assign a class label to each cluster. They then use KNN to classify a new instance by finding the closest cluster to the data point. However, our implementation of this method differs as in their research, they used three random k-values for the number of clusters and looked at the performances of each. In our implementation, we implement the reverse elbow method to find an optimal k using the validation dataset before testing the performance with the testing dataset.

III. Dataset and Features

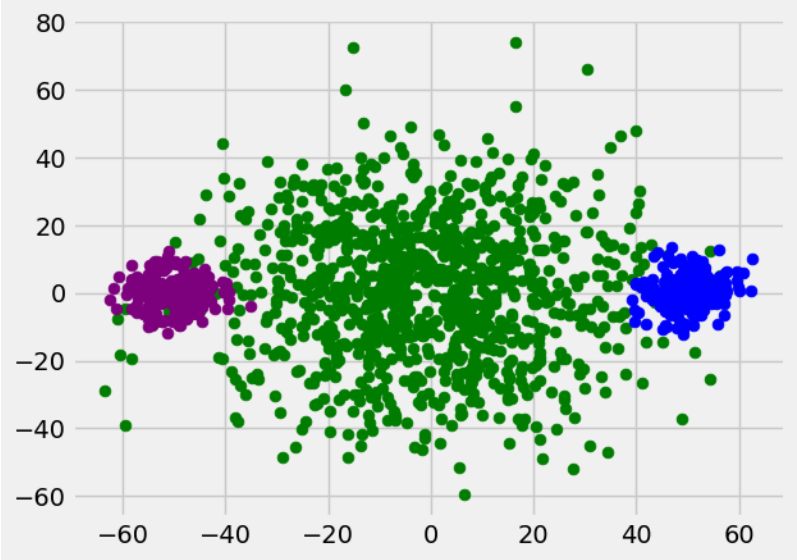
We are using a dataset generated using the make_blobs function in the sci-kit learn library. This dataset has a total of 1400 instances. The train-validation-test split ratios are 70 to 15 to 15. This dataset has two features, an x-coordinate and a y-coordinate. There are three class values and the class distributions are summarized in Table 1.

Table 1: Data Summary

Class Value	Number of Instances
0	1000
1	200
2	200

Since this dataset was generated using sci-kit learn, no preprocessing or normalization was necessary. Figure 1 is a visualization of the dataset generated by sci-kit learn. Green points represent instances with a class value of 0, blue points represent instances with a class value of 1, and purple points represent instances with a class value of 2.

Figure 1: Dataset Visualization



IV. Methods

KNN is a supervised machine learning classification algorithm that classifies an instance by computing the distance between the instances' attribute values and those of all other instances in the dataset. The distance formula used is commonly the Euclidean distance formula, although other distance formulas may also be used, such as cosine similarity and hamming distance (qualitative data). After calculating the distances, they will be ranked from least to greatest and the first k distances will be used to classify the instance. The labels that correspond to the first k distances will need to be found, and the instance will be classified by the majority of the first k labels.

Euclidean Distance Formula:

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

K-Means is an unsupervised machine learning classification algorithm for quantitative data. It classifies an instance by creating clusters within the data and finding the closest cluster to the instance. The instance is then classified as belonging to that cluster. Because K-Means is an unsupervised machine learning classification algorithm, it cannot classify the instance as a specific label. The creation of clusters work by assigning random centroids in the data and using the Euclidean distance formula between data points and the centroid to adjust the centroid until the cluster converges.

We want to resolve the limitations of KNN and K-Means as mentioned in the introduction by combining the two together. First, we will use K-Means to make clusters in the data. To do so, we will use Sci-kit Learn's implementation of K-Means with the Euclidean distance formula to formulate clusters and determine locations of centroids. Then, we will find

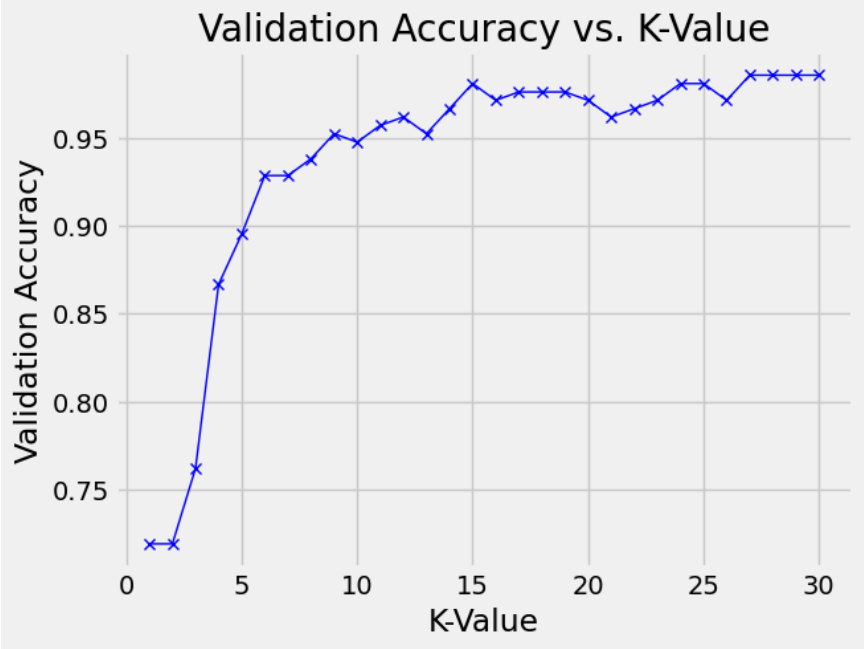
the majority label in each cluster using majority voting and assign the centroid of each cluster to the determined label. When classifying an instance, we will use our own implementation of KNN with k=1, calculating the distance between the instance to be classified and the centroids of the clusters. Therefore, the closest centroid's label will be the classification of the instance.

A factor that we took into account was finding the optimal k value for K-Means to determine the best number of clusters to increase the accuracy of our model. We do so by plotting the accuracies of the model with different clusters in a range determined by the user. In our instance, we used k values from 1 to 14. Then, we examine the resultant graph to find the lowest k-value at which high accuracy is achieved as to lessen the time complexity of our model.

V. Results and Discussion

For K-Closest Clusters (KCC), the k-value represents the total number of clusters that are to be created by the KCC classifier. The chosen value of k was 15, which was determined by evaluating the accuracy of each KCC classifier with a different k-value on the validation set for each possible value of k. We then graphed the accuracies, shown in Figure 2, that each model attained and manually selected the k-value to use on the test dataset.

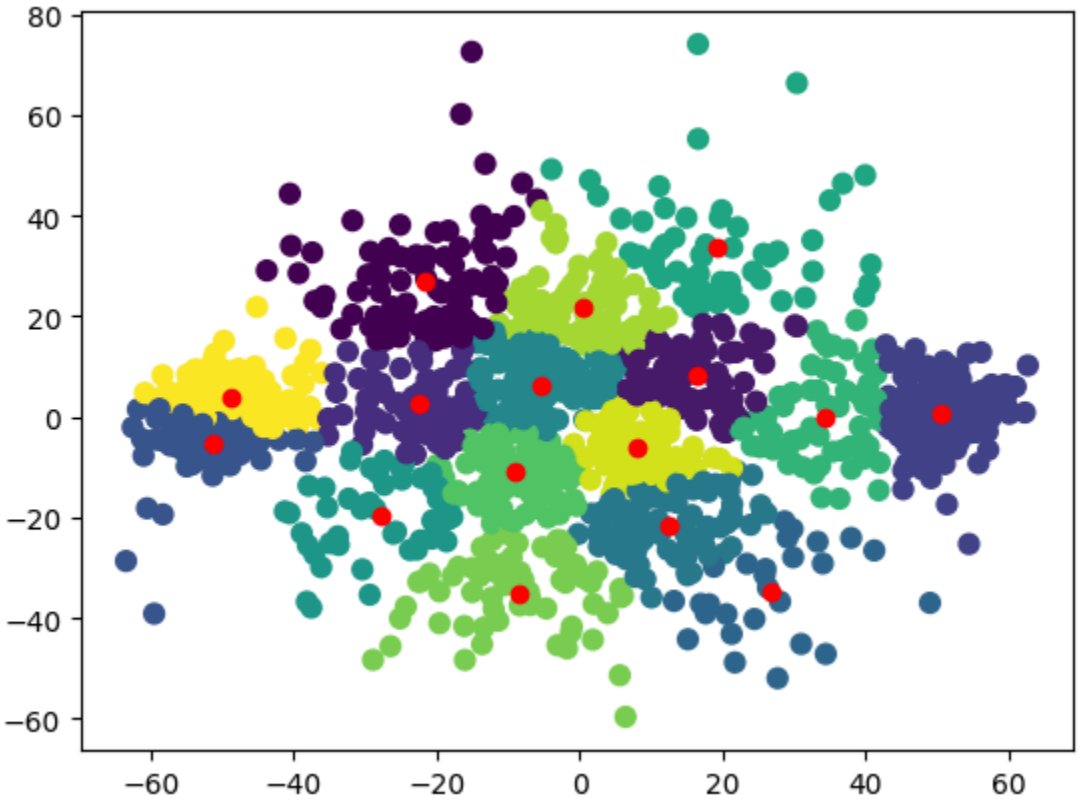
Figure 2: Validation Accuracy with respect to K-value



The rationale for choosing the k-value is to balance accuracy and time complexity. A high k-value will achieve a higher accuracy, but will cause classification to be more computationally expensive. A low value of k may cause KCC to underfit on the data. Thus, we attempted to choose a k-value to minimize computation time without compromising accuracy.

After choosing a k-value of 15, the KCC algorithm created clusters that are depicted in Figure 3. The red points denote the centroids of the clusters, which are used for classification.

Figure 3: Cluster and Centroid Visualization



We tested the KCC model on the test set and the results we achieved compared to KNN and the Nearest Cluster algorithm from [1] are displayed in Table 2.

Table 2: Results Summary

Metric	KNN	Nearest Cluster (K=3)	Nearest Cluster (K=5)	Nearest Cluster (K=7)	KCC
Accuracy	95.7%	71.9%	92.9%	93.8%	96.2%
Precision	0.922	0.610	0.964	0.965	0.931
Recall	0.979	0.731	0.880	0.893	0.981
Classification Time	0.229s	0.003s	0.003s	0.003s	0.007s

From these results, we can observe that KCC has achieved a slightly higher accuracy, precision, and recall than KNN. The greatest advantage of KCC in comparison to KNN lies in the classification time. To classify the 200 instances in the test set, KCC took 0.007 seconds, which is approximately 32.7 times faster than the time KNN took to classify the instances.

The reason for this significant difference in classification time is that KCC creates a model, while KNN does not create a model. To classify an instance, KNN must calculate the distance between every point in the training set with the given instance. For datasets with a large number of instances, KNN can become very computationally expensive. In contrast, our dataset creates a model that assigns a class value to every cluster centroid, so to classify an instance, the KCC algorithm must complete k distance calculations, which can be significantly more efficient than KNN, given that an optimal k-value is selected.

Additionally, KCC performed better in terms of accuracy than the Nearest Cluster models with K=3, K=5, and K=7. KCC also had a better recall value than the Nearest Cluster models with K=3, K=5, and K=7, indicating that KCC classifies instances that overlap between the clusters better than Nearest Cluster. By optimizing the number of clusters for the specified dataset, the KCC algorithm delivers a significantly higher test accuracy than Nearest Cluster.

VI. Conclusion and Future Work

In this paper, we develop a novel machine learning algorithm that employs clustering techniques from K-means to improve the K-Nearest Neighbors algorithm. Future work regarding this method could investigate using density clusters such as DBSCAN instead of distance based clustering algorithms like K-Means. Additionally, another idea for future research includes making the process of determining optimal k value more efficient and effective. More research could go into calculating an equation to find the upper threshold of k values to be tested. The process for determining the actual optimal k value by looking at the k value vs. accuracy graph may be automated, instead of requiring a human user to determine the optimal k-value.

VII. Contributions

Dataset preprocessing: Julia
Algorithm Development: Julia
Introduction: Yuan
Related Work: Julia and Yuan
Dataset and Features: Julia
Methods: Yuan
Results and Discussion: Julia
Conclusion and Future Work: Yuan

References

[1] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, “Efficient k NN classification algorithm for big data,” Neurocomputing, vol. 195, pp. 143–148, Jun. 2016, doi: https://doi.org/10.1016/j.neucom.2015.08.112.

[2] Levent Ertoz, M. Steinbach, and V. Kumar, “Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data,” May 2003, doi: https://doi.org/10.1137/1.9781611972733.5.

[3] Alizadeh, “A New Method for Improving the Performance of K Nearest Neighbor using Clustering Technique,” Journal of Convergence Information Technology, vol. 4, no. 2, Jun. 2009, doi: https://doi.org/10.4156/jcit.vol4.issue2.alizadeh.