Reducing the Number of Calculations in k-nn by Class Representatives AtB Voting

Antoni Jaszcz[a]

[a]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND

Abstract

The problem of clustering and classification is a key task in the field of artificial intelligence. Due to the increasing amounts of data in the digital world, existing solutions must be quickly adapted to them. Unfortunately, some solutions have quite big limitations. An example of what is *k*-nearest neighbors (*k*-nn). In this paper, we propose two improvements, which can be used for increasing the accuracy of used tools and decrease the operating time. Described modification focus on introducing class representatives and voting mechanism among the best data. Voting is carried out on samples that have achieved some degree of being the best, such as above the average distance among all of them using the Euclidean metric. The proposed solutions are tested and analyzed against the original version of *k*-nn.

Keywords

Data classification, knn, Voting technique

1. Introduction

The growth of the digital world comes down to increasing the amount of information. All data is most often analyzed and used in various practical applications. Hence, a very important problem is the method of searching, analyzing, and classifying the obtained data. A very important element and operation is modeling new solutions, as well as modifying the existing ones.

Recent years have shown that the use of intelligent tools such as neural networks [1, 2], *k*-nn [3] has a great practical application in the system of ship's monitoring [4, 5] and Internet of Things solutions [6, 7, 8]. Newly entered data in any system must be quickly tagged or marked with some classes for further processing. Here, the most important parameters are not only the accuracy of the tool but also the time of processing the new samples. In this case, when huge amounts of new information are obtained, the long processing time may result in queuing and even overloading the system. This work focuses on the analysis and modification of the classical *k*-nn clustering algorithm. The main contribution of this paper are:

- improving *k*-nn by analyzing only the representatives of classes.
- analyzing the *k* value by voting only through the best samples.

SYSTEM 2021 @ Scholar's Yearly Symposium of Technology, Engineering and Mathematics. July 27–29, 2021, Catania, IT

aj303181@student.polsl.pl (A. Jaszcz)

2. Related works

Despite the passage of years, *k*-nn is still used and improved. This is evidenced by the latest research results published by scientists from around the world. An example of this can be shown in [9], where an algorithm was used but with modification to only one-class rule and with adaptive Mahalanobis-squared distance. A similar idea was presented in [10], where *k*-nn was combined with a meta-heuristic algorithm inspired by the cuckoo search idea. This solution was analyzed in a multidimensional knapsack problem and the obtained results present great potential in such an approach. Moreover, there are ideas of using clustering method in some classifiers like SVM, CNN, etc. [11, 12, 13, 14]. Except for the mentioned disadvantages like time computational, there is an aspect of security which was examined by research in [15], where the practical differential privacy was shown. Again, in [16], the multi-dimensional classification with augmentation method using *k*-nn was presented. Other latest modifications and hybridization are optimizing this tool by many systems [17].

An example of using *k*-nn is traffic congestion monitoring system [18, 19, 20], or mapping the leaf area index [21]. Another important application area is organic solar cells defects classification [22], or even recommendation systems [23, 24].

3. *k*-nn algorithm in machine learning

In artificial intelligence, the *k*-nn algorithm is used to assess new objects according to those in the provided database. The class of such objects consists of a set of parameters and the boolean-type outcome. Based on the

parameters, the algorithm chooses k objects (referred to like the neighbors) from the database and appropriately to their outcome, evaluates those new objects.

For every new object to assess, the process can be divided into 4 steps:

- Calculating the Euclidean distance between the object and each one existent in the database.
- Sorting the database by the obtained length.
- Choosing k neighbors.
- Voting among the neighbors.

The main idea behind this algorithm is to consider k objects that are most familiar to the one being evaluated, so those k-objects that are closest to the assessed one in p-dimensional space (where p stands for the number of appraised parameters of the class). The distance is calculated by the euclidean distance formula given below. Euclidean metric:

$$d(A, B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2} \tag{1}$$

where
d - distance between two points
n - number of dimensions in which two points are defined
A, B - two points in n-dimensional space
A_i, B_i - coordinate values of two points in i-dimension

Having calculated the distances and sorted the data by their values, the algorithm carries out the voting among k-chosen neighbors. The outcomes of the neighbors are counted up and the one in the majority wins resulting in the new object being given a corresponding outcome.

From mathematical point, the k-nn can be defined as probability estimator, where the posteriori membership of p sample to k class is calculated as:

$$\hat{p} = \frac{1}{K}\sum_{i=1}^{n} I(d(p, p_i) \leqslant (p, p^k))I(y_i = k) \tag{2}$$

where $p^{(k)}$ is K-th point from the training sample x_p and $k = 1, \cdots, L$. Then, the k-nn classifier has a form of:

$$\hat{d}_{KNN}(p) = \underset{k}{argmax}\hat{p}(k|p) \tag{3}$$

The algorithm raises two major concerns:

1. Time complexity. The complexity of the k-nn algorithm is.
$$O(n^2 log\, n)$$
where n is the number of elements in the database. Furthermore, the accuracy of the algorithm depends highly on the number of objects. The more objects we account for, the higher the chance of finding more familiar ones to the one being assessed.
2. Choosing k.
There is no one way to decide how many neighbors should be considered. There are many approaches to do so. In this paper, some of them are described and compared.

4. Proposition

4.1. Finding most accurate *k* for given database

The basic approach is to choose that k which is most accurate for the provided database. The process of finding the k need not be performed more than once per database. The algorithm of that is as follows:

Algorithm 1: *finding the most accurate k value*

```
   Input: database of objects
1  Start;
2  Divide the input into two groups:
     validation-group and training-group in set
     proportion;
3  foreach object in validation-group do
4  |  foreach object in training-group do
5  |  |  Calculate the Euclidean distance;
6  |  end
7  |  Sort training-group by distance;
8  |  foreach k value (ranging from 1 up to the
   |    number of objects in testing-group) do
9  |  |  Vote among the neighbours;
10 |  |  if the outcome matches that of the object
   |  |    from training-group then
11 |  |  |  correct[k] = correct[k] + 1;
12 |  |  end
13 |  end
14 end
15 return correct[];
16 Stop;
```

In the algorithm above, *correct* variable is an array containing integer values indicating the number of objects from the validation group which was correctly assessed, for each k being considered.

Having the array as an output, we can now find k value with the most properly assessed objects, so the k we are looking for. By finding the greatest value in the array, k value is its corresponding position in the

array. Furthermore, we can calculate the accuracy by the following formula:

$$accuracy = \frac{TP + TN}{TP + TN + FP * FN} \tag{4}$$

where *TP* means true positive, *TN* true negative, *FN* falsenegative and *FP* false positive.

4.2. The mean method

Another approach aims not to find one universal *k*, but rather have it calculated while assessing every new record. In the mean method, only those neighbors have considered whose distance is no greater than the mean distance. The pseudo-algorithm of the method is presented in Alg. 2.

Algorithm 2: *the mean method*

```
   Input: database of objects
1  Start;
2  Divide the input into two groups:
     validation-group and training-group in set
     proportion;
3  foreach object in validation-group do
4  |  mean-distance = 0;
5  |  foreach object in training-group do
6  |  |  Calculate the Euclidean distance;
   |  |    mean-distance = mean-distance +
   |  |    object[t].distance;
7  |  end
8  |  Sort training-group by distance;
9  |  foreach object in training-group do
10 |  |  if object[t].distance > mean-distance then
11 |  |  |  break;
12 |  |  else
13 |  |  |  if the outcome matches that of the
   |  |  |    object from training-group then
14 |  |  |  |  correct = correct + 1;
15 |  |  |  end
16 |  |  end
17 |  end
18 end
19 return correct;
20 Stop;
```

In the algorithm above, *correct* is an integer variable indicating the number of objects from the validation group which was correctly assessed. The $object_t \cdot distance$ is the distance between the object from the training group and the one from validation-group, that is currently being considered in the loop.

4.3. Reducing the size of the database only to the representative objects

The last method examined in this paper is partially a modification of the first method. The main goal of this method is to transform the database in such a way that will not affect the effectiveness of the algorithm itself and at the same time will improve its time complexity.

5. Experiments

All tests were carried out on a computer with the following specifications:
Processor: Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz 3.30GHz
Installed RAM: 24GB
System Type: 64-bit operating system, x64-based processor
The database used in the experiments consists of data of 768 patients defined by the 8 following parameters:

1. No. pregnancies.
2. Glucose level.
3. Blood pressure.
4. Skin thickness.
5. Insulin level.
6. BMI.
7. Diabetes Pedigree Function.
8. Age.

Each object is also given a boolean outcome, indicating whether the patient is diagnosed with diabetes or not. In the following results, the time efficiency and accuracy are compared. The accuracy is calculated by the following formula:

$$a(k) = V_k \div V_s \tag{5}$$

where $V_k \leqslant V_s$
a - accuracy [%]
k - given *k* value
V_k - number of the objects from validation-group properly assessed for a given *k*
V_s - number of objects in validation group

In this paper, the results were rounded to ones.

5.1. Algorithm 1:

In Fig 1, relation between *k* value and obtained accuracy is presented. The visible plateau forming after *k* value exceeds 155, is a result of organising the database beforehand in such a manner, so it would give that specific result. By doing so, we were assured that the algorithm did not contain unexpected errors.
In table 1, top 10 most accurate *k* values are displayed.

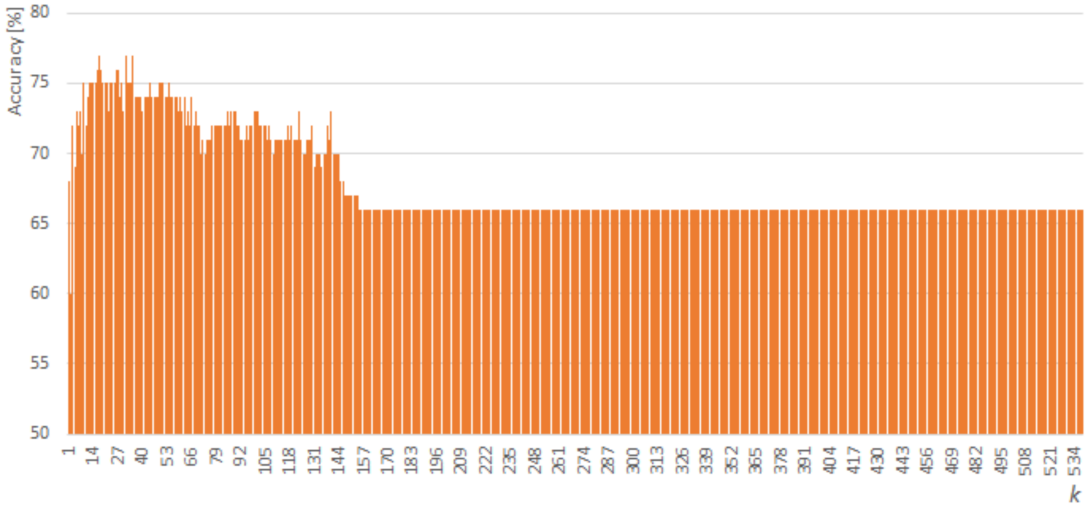

Figure 1: Accuracy(k)

Table 1
Sample value

k	Accuracy [%]
17	77
31	77
35	77
16	76
18	76
26	76
27	76
9	75
12	75
13	75

5.2. Algorithm 2:

In this method, the k value is constantly changing. There is only one result regarding the whole validation group: accuracy equal to 66%. It is also worth mentioning, that in this case, the accuracy formula changes slightly. Instead of using V_k, we use V_c, which stands for the number of objects validation-group correctly assessed. The formula changed ever so slightly is presented then as follows:

$$a(k) = V_c \div V_s \tag{6}$$

5.3. Algorithm 3:

As shown in Fig 2, the number of objects in the training group and as a result, k values to consider is much

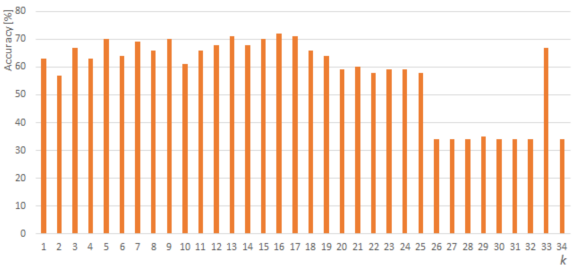

Figure 2: Accuracy(k) - algorithm C

lower. That is because the size of the reformed training group becomes dependant on the number of parameters we consider, as well as the number of different possible outcomes the object may have. The size of the group is as follows:

$$T_s(N_p, N_O) = N_O \cdot (2 \cdot N_p + 1) \tag{7}$$

where

T_s - size of the training group
N_p - number of parameters being considered
N_O - number of possible outcomes

In our example, we considered 8 parameters and 2 possible outcomes for every patient, resulting in the size of the new training group being equal to 34. As a result, there was a significant improvement in the number of calculations to perform. Furthermore, the most accurate this method could get was 72% (for the k value of 16), being only 5 percentage points lower than in the first

Table 2
Average accuracy and mean execution time.

Method	Top accuracy[%]	Mean exec. time [ms]
A	75	2597
B	66	2599
C	72	140

method.

In the table 2, top accuracies and mean execution times (obtained from 100 tries) are shown.

6. Conclusions

When analyzing results, a couple of things become apparent. First, that the mean method improves neither accuracy nor the time efficiency of the *k*-nn algorithm. Second, that modifying a given database greatly improves time efficiency, without significant accuracy loss. It is worth repeating, that the sole effectiveness of *k*-nn algorithm depends highly on the number of testing objects, so on the size of the database, we are given. Having that in mind, any time-complexity improvement should be beneficial when provided much more data. The ineffectiveness of the mean method can be further explained by the fact, calculating mean distances is not a particularly precise way of choosing the neighbors. The basic method, providing accuracy for a range of *k* values produces much more precise insight into the objects in a given database. Because of that, we suggest using that method over the mean one, when it comes to choosing *k*.

The last method, being partially a modification of the basic approach, should be chosen as a leading implementation of the *k*-nn algorithm. Yet, it can still be improved greatly. Should there be more parameters or possible outcomes, the time efficiency can be vulnerable. However, we believe that by reducing the number of parameters only to those significant, such problem should be omitted. Nevertheless, creating class representatives is a great way to improve the *k*-nn algorithm and it creates a base for further optimizations.

References

[1] N. Tuccitto, A. Bombace, A. Torrisi, A. Licciardello, G. Lo Sciuto, G. Capizzi, M. Woźniak, Probabilistic neural network-based classifier of tof-sims single-pixel spectra, Chemometrics and Intelligent Laboratory Systems 191 (2019) 138–142.

[2] G. Lo Sciuto, G. Capizzi, S. Coco, R. Shikler, Geometric shape optimization of organic solar cells for efficiency enhancement by neural networks, Lecture Notes in Mechanical Engineering (2017) 789–796. doi:10.1007/978-3-319-45781-9_79.

[3] A. Fan, C. Gardent, C. Braud, A. Bordes, Augmenting transformers with knn-based composite memory for dialog, Transactions of the Association for Computational Linguistics 9 (2021) 82–99.

[4] D. Połap, M. Włodarczyk-Sielicka, N. Wawrzyniak, Automatic ship classification for a riverside monitoring system using a cascade of artificial intelligence techniques including penalties and rewards, ISA transactions (2021).

[5] D. Polap, M. Wlodarczyk-Sielicka, Classification of non-conventional ships using a neural bag-of-words mechanism, Sensors 20 (2020) 1608.

[6] G. Capizzi, C. Napoli, S. Russo, M. Woźniak, Lessening stress and anxiety-related behaviors by means of ai-driven drones for aromatherapy, volume 2594, 2020, pp. 7–12.

[7] A. Zielonka, A. Sikora, M. Woźniak, W. Wei, Q. Ke, Z. Bai, Intelligent internet of things system for smart home optimal convection, IEEE Transactions on Industrial Informatics 17 (2020) 4308–4317.

[8] R. Avanzato, F. Beritelli, M. Russo, S. Russo, M. Vaccaro, Yolov3-based mask and face recognition algorithm for individual protection applications, in: CEUR Workshop Proceedings, 2020, pp. 41–45.

[9] H. Sarmadi, A. Karamodin, A novel anomaly detection method based on adaptive mahalanobis-squared distance and one-class knn rule for structural health monitoring under environmental effects, Mechanical Systems and Signal Processing 140 (2020) 106495.

[10] J. García, C. Maureira, A knn quantum cuckoo search algorithm applied to the multidimensional knapsack problem, Applied Soft Computing 102 (2021) 107077.

[11] D. Połap, An adaptive genetic algorithm as a supporting mechanism for microscopy image analysis in a cascade of convolution neural networks, Applied Soft Computing 97 (2020) 106824.

[12] M. Tanveer, A. Sharma, P. N. Suganthan, Least squares knn-based weighted multiclass twin svm, Neurocomputing 459 (2021) 454–464.

[13] S. Liao, G. Li, J. Li, D. Jiang, G. Jiang, Y. Sun, B. Tao, H. Zhao, D. Chen, Multi-object intergroup gesture recognition combined with fusion feature and knn algorithm, Journal of Intelligent & Fuzzy Systems 38 (2020) 2725–2735.

[14] Z. Bai, Y. Li, M. Woźniak, M. Zhou, D. Li, Decomvqanet: Decomposing visual question answering deep network via tensor decomposition and regression, Pattern Recognition 110 (2021) 107538.

[15] Y. Zhu, X. Yu, M. Chandraker, Y.-X. Wang, Private-knn: Practical differential privacy for computer vision, in: Proceedings of the IEEE/CVF Conference

on Computer Vision and Pattern Recognition, 2020, pp. 11854–11862.
[16] B.-B. Jia, M.-L. Zhang, Multi-dimensional classification via knn feature augmentation, Pattern Recognition 106 (2020) 107423.
[17] J. Hu, H. Peng, J. Wang, W. Yu, knn-p: A knn classifier optimized by p systems, Theoretical Computer Science 817 (2020) 55–65.
[18] F. Harrou, A. Zeroual, Y. Sun, Traffic congestion monitoring using an improved knn strategy, Measurement 156 (2020) 107534.
[19] C. Napoli, G. Pappalardo, E. Tramontana, Improving files availability for bittorrent using a diffusion model, in: 2014 IEEE 23rd International WETICE Conference, IEEE, 2014, pp. 191–196.
[20] G. Borowik, M. Woźniak, A. Fornaia, R. Giunta, C. Napoli, G. Pappalardo, E. Tramontana, A software architecture assisting workflow executions on cloud resources, International Journal of Electronics and Telecommunications (2015).
[21] F. Jiang, A. R. Smith, M. Kutia, G. Wang, H. Liu, H. Sun, A modified knn method for mapping the leaf area index in arid and semi-arid areas of china, Remote Sensing 12 (2020) 1884.
[22] G. Lo Sciuto, G. Capizzi, R. Shikler, C. Napoli, Organic solar cells defects classification by using a new feature extraction algorithm and an ebnn with an innovative pruning algorithm, International Journal of Intelligent Systems 36 (2021) 2443–2464.
[23] C. Napoli, G. Pappalardo, E. Tramontana, Using modularity metrics to assist move method refactoring of large systems, in: 2013 Seventh International Conference on Complex, Intelligent, and Software Intensive Systems, IEEE, 2013, pp. 529–534.
[24] S. G. K. Patro, B. K. Mishra, S. K. Panda, R. Kumar, H. V. Long, D. Taniar, I. Priyadarshini, A hybrid action-related k-nearest neighbour (har-knn) approach for recommendation systems, IEEE Access 8 (2020) 90978–90991.