Handwritten digit recognition by average-representatives clustering

Szymon Hankus[1,*]

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

Abstract

K-Nearest Neighbours algorithm had been a go-to method for multi-parameter classification problems. This changed with the introduction of neural networks and more sophisticated solutions. Despite that, k-NN to this day is a reliable solution for addressing problems of clustering due to its simplicity and effectiveness. This, however, is not without its downsides. High computation complexity and a large quantity of data required to make the method effective have made it less and less applicable to modern problems. However, new adjustments and unusual applications for the algorithm can be found. In this paper, I propose using the k-NN method for handwritten digit recognition, with average-representative methods as a time-optimization tool.

Keywords

k-nn, clustering, representatives, digit recognition

1. Introduction

The growth of data and fast-computation requirements have made old technologies obsolete. Or so it may seem. These methods are still often used today [1, 2], as they are simple, reliable and accurate. Furthermore, they are worth revisiting, as they can still be improved and optimized.

The main problems with k-nn method are its large computational complexity and large quantity of data required to make the algorithm perform well. This can be a major setback, especially in systems that highly depend on speed, such as SQL servers [3]. For years now, different approaches have been designed to address this issue. Many of them rely on reducing the number of looked-upon samples during algorithm-predicting calculations or using data processing tools [4, 5]. In [6] the author proposes a solution based on considering only representatives of the class. This way redundant data can be removed, without loss of data potency [7].

Another issue is the infamous curse of dimensionality. This problem is not only related to clustering but also to modern technologies, such as Neural Networks, Convolutional Neural Networks [8] and Recurrent Neural Networks [9]. That is why it is important to find new solutions in older models, which can be applied to newer once to resolve issues, which other ways can be easily omitted due to the complexity of the methods [10].

K-nn is widely recognized as a method for solving clustering problems, such as recommendation systems. However, the method has been rediscovered time and time again as a solution for unusual problems, such as surveillance [11], cyber-security [12] and image detection [13]. Hence it is important to keep seeking new solutions, which very often can be then used for improving new, edge technologies.

2. Methodology

2.1. k-NN algorithm

k-NN is a simple, well-performing algorithm used in statistics for classification and regression. The principle of the algorithm is as follows: given the dataset of N samples, pre-labeled as one of the C classes, with P numeric parameters, the algorithm gives a prediction for the new sample, by considering its k 'nearest neighbors' (points in the $dim(P)$ dimensional metric space, least further away) and returning the adequate output:

- **Classification:** vector of class membership, from which the assessed class is chosen during a plurality voting among the neighbors.
- **Regression:** vector of mean parameters, calculated from the neighbors.

The prediction process can be divided into four steps:

1. Calculating the distances between the new sample and each one from the initial dataset.
2. Sorting samples accordingly to obtained distance.
3. Choosing k samples (neighbors).
4. Performing voting among the neighbors and returning the final prediction.

Proces przewidywania można podzielić na cztery etapy:

8th Symposium for Young Scientists in Technology, Engineering and Mathematics (SYSTEM 2023), Poland, May. 15-17, 2023

*Corresponding author.

sh303175@student.polsl.pl (S. Hankus)

1. Obliczanie odległości między nową próbką a każdą próbką z początkowego zbioru danych.
2. Sortowanie próbek zgodnie z uzyskaną odległością.
3. Wybór k próbek (sąsiadów).
4. Przeprowadzenie głosowania wśród sąsiadów i zwrócenie ostatecznej prognozy.

Most often Euclidean norm is chosen for the distance metric in P-dimensional space. Because of this, the initial normalization of the parameters is crucial for the proper functionality of the algorithm.

The algorithm relies highly on the number of samples. The more samples k-NN has to work with, the better chance of choosing similar objects, resulting in better certainty of predicted class and hence, generally better accuracy. Unfortunately, this creates the following problems:

- Any anomalies or data poisoning are highly disruptive.
- High computational complexity and a large quantity of data required for proper functioning result in relatively high time complexity, which makes it hard to implement in real-time scenarios.

2.2. Average-representatives

To tackle the issue connected with the high computational time mentioned in Sec. 2.1, I propose the method of reducing samples in the database without losing its informational value, based on sample-averaging. Given a number of samples n_{c_i} of a certain class c_i, we can transform them into $n_{samples}$ 1 samples, each formed from n_{layer} 2 consecutive samples of the class (except for the last one, which is made from n_{left} 3 leftover samples), by averaging them. This is further presented with an example in Sec. 3.2. The process of creating j-th average-representative $\hat{x}_{i_j}$ of i-th class is presented in eq. 4.

$$n_{samples} = \lceil \log_2 n_{c_i} \rceil \tag{1}$$

$$n_{layer} = \lceil \frac{n_{c_i}}{n_{samples}} \rceil \tag{2}$$

$$n_{left} = n_{c_i} - ((n_{samples} - 1) * n_{layer}) \tag{3}$$

$$\hat{x}_{i_j} = \begin{cases} \frac{\sum_{k=1}^{n_{layer}} x_k}{n_{layer}} & , for \quad j < n_{samples} \\ \frac{\sum_{k=1}^{n_{left}} x_k}{n_{left}} & , for \quad j = n_{samples} \end{cases} \tag{4}$$

2.3. Reducing number of parameters by feature-projection

In order to reduce the number of dimensions, the projection of a set of related parameters into a new feature can be applied. The product of such an operation is a new set of parameters, representing newly created features. This can be achieved by, for example, averaging a subset of related parameters P' into a new singular parameter $\hat{p}'$ (presented as a numerical value). Such operation can be shown as:

$$\hat{p}' = \frac{\sum_{i=1}^{dim(P')} p'_i}{dim(P')} \tag{5}$$

3. Experiments

In this section, all conducted experiments are described.

3.1. Data and training setting

In my research, I used [14]UCL Handwritten Digits Data Set, a well-known dataset for optical handwritten-digit recognition. It consists of 1797 samples of 8x8 pixel grayscale digit images (with pixel intensity in the range of [0,16]). Each class contains roughly 180 samples. Class examples (from 0 to 4) are shown in Fig. 1.

Because all values considered by the algorithm are in the range [0,16], the crude dataset is already normalized.

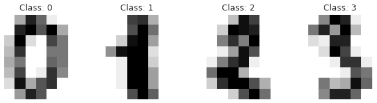

Figure 1: Example data samples

3.2. Image representatives

In the experiments, the impact of applying the average-representative method described in Sec. 2.2 was tested, by transforming training samples into an adequate number of representatives for each class. In other words, such representatives were created by overlapping a certain number of samples of a given class and then dividing the product by their quantity. This is best illustrated in Fig. 2.

3.3. Column projection

As described in Sec. 2.3, new features were created by averaging related parameters. In this case, I considered pixels in the same column as related parameters. By averaging pixel values in each column, the number of

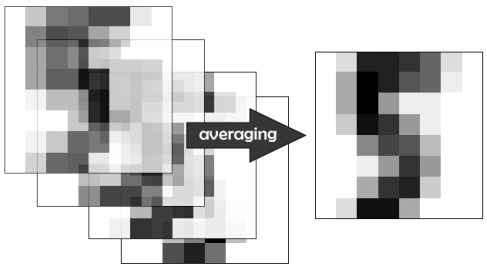

Figure 2: Creating representatives from samples of class 5

parameters was reduced from $8x8 = 64$ to only $8x1 = 8$. This, however, had to be done to samples in the training set, as well as any sample that was being predicted.

3.4. Choosing k

Another issue connected with the algorithm is choosing the right k parameter (number of considered neighbors). In order to choose it efficiently, I tested the performance of each proposed model with different k parameters, in the range from 1 to 15, and decided on k based on obtained model accuracy. Each model was trained and tested on the subsets, divided 3:1 from the original dataset. Training and testing sets were the same for each model. The accuracy obtained for different k values is shown in Fig. 3-5. Based on the obtained results, chosen k parameters were:

- 7 for base k-nn model
- 2 for k-nn model with implemented average-representatives method
- 2 for k-nn model with feature-projection method

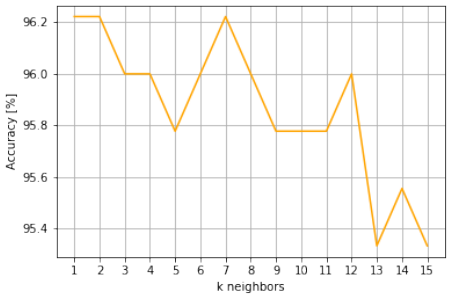

Figure 3: Dependence of accuracy of base k-nn model on different k values

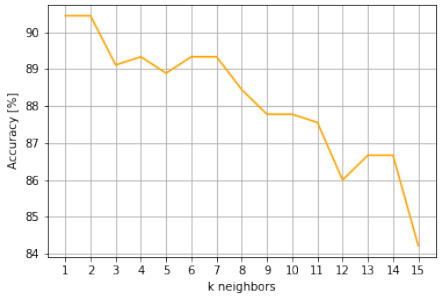

Figure 4: Dependence of accuracy of k-nn model with implemented average-representatives method on different k values

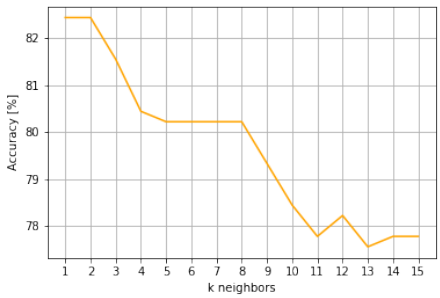

Figure 5: Dependence of accuracy of k-nn model with implemented feature-projection method on different k values

4. Results

To properly assess the performance of the models, 4-fold cross-validation was conducted. The models were evaluated with four metrics:

- Accuracy
- Precision (macro avg.)
- Recall (macro avg.)
- f1-score (macro avg.)

Calculated metrics are displayed in Tab. 1-4.

Table 1
Models' accuracy score

Model	fold 1	fold 2	fold 3	fold 4
base k-nn	96.44	95.55	97.33	96.21
		avg = 96.38		
representatives	91.76	91.54	93.76	90.42
		avg = 91.87		
projection	73.50	75.95	81.74	82.63
		avg = 78.45		

Table 2
Models' precision score (macro)

Model	fold 1	fold 2	fold 3	fold 4
base k-nn	96.54	95.58	97.40	96.25
		avg = 96.44		
representatives	92.44	92.04	93.93	90.72
		avg = 92.28		
projection	74.63	76.03	82.15	82.59
		avg = 78.85		

Table 3
Models' recall score (macro)

Model	fold 1	fold 2	fold 3	fold 4
base k-nn	96.43	95.53	97.39	96.14
		avg = 96.37		
representatives	91.69	91.47	93.89	90.41
		avg = 91.87		
projection	73.83	75.67	81.93	82.40
		avg = 78.46		

Table 4
Models' f1-score (macro)

Model	fold 1	fold 2	fold 3	fold 4
base k-nn	96.41	95.52	97.39	96.10
		avg = 96.36		
representatives	91.81	91.50	93.88	90.32
		avg = 91.88		
projection	73.54	75.54	81.82	82.13
		avg = 78.26		

Table 5
Models' computational time comparison for conducted cross-validation

Model	computational time
base k-nn	19.875s
representatives	1.250s
projection	19.531s

To further visualize obtained results, confusion matrices were created for each fold, and their averages are shown in Fig. 4-8. Comparison of mean accuracy of proposed models is additionally displayed in Fig. 4.

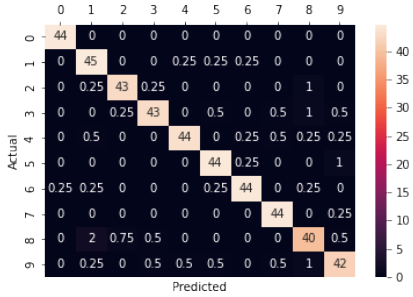

Figure 6: Mean confusion matrix of the base model

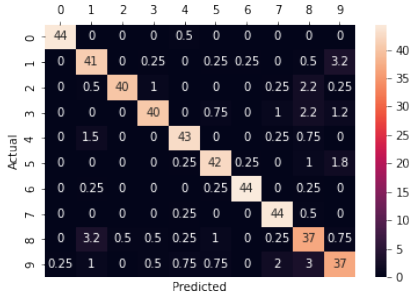

Figure 7: Mean confusion matrix of the model with implemented average-representatives

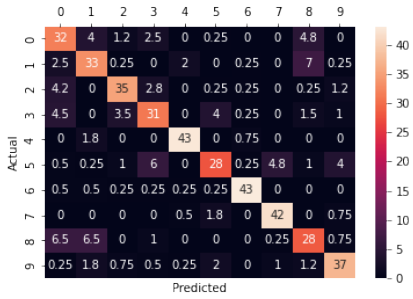

Figure 8: Mean confusion matrix of the model with implemented feature-projection

5. Conclusion

As can be seen in Tab. 1, k-nn model performed exceptionally well with a given task. Not only accuracy but all other measured metrics presented in Tab. 2-4 remain strong. For such a simple model, the result is astonishing. However, as mentioned in Sec. 2.1, one of the biggest problems is its computational complexity, resulting in high calculation time along with data growth. This issue can be addressed by the implementation of average representatives. As can be observed in Tab. 5, applying this method immensely reduces the computational time required to predict new samples. However, it is not without consequences. As can be drawn out from the Tab. 1, the representatives-method performed roughly 4.5 percentage points worse in terms of accuracy, than the base model. Although quality compromise for speed

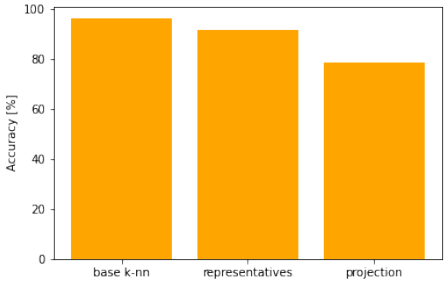

Figure 9: Mean accuracy comparison of the models

is unavoidable, the increase in speed far outweighs the decrease in accuracy and the method may be used as a valid solution, where speed is paramount. The projection method did rather poorly in terms of both speed and accuracy. Feature-projection method can be used as a valid solution for reducing data dimensional, even tough in this case it turned out futile. Reducing the number of parameters neither significantly improved speed nor accuracy.

References

[1] M. Ulinskas, M. Woźniak, R. Damaševičius, Analysis of keystroke dynamics for fatigue recognition, in: O. Gervasi, B. Murgante, S. Misra, G. Borruso, C. M. Torre, A. M. A. Rocha, D. Taniar, B. O. Apduhan, E. Stankova, A. Cuzzocrea (Eds.), Computational Science and Its Applications – ICCSA 2017, Springer International Publishing, Cham, 2017, pp. 235–247.

[2] A. Jaszcz, D. Połap, Aimm: Artificial intelligence merged methods for flood ddos attacks detection, Journal of King Saud University-Computer and Information Sciences 34 (2022) 8090–8101. URL: https://www.sciencedirect.com/science/article/pii/S1319157822002580. doi:https://doi.org/10.1016/j.jksuci.2022.07.021.

[3] B. Yao, F. Li, P. Kumar, K nearest neighbor queries and knn-joins in large relational databases (almost) for free, in: 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010), 2010, pp. 4–15. doi:10.1109/ICDE.2010.5447837.

[4] D. Połap, M. Woźniak, R. Damaševičius, R. Maskeliūnas, Bio-inspired voice evaluation mechanism, Applied Soft Computing 80 (2019) 342–357.

[5] M. Woźniak, A. Sikora, A. Zielonka, K. Kaur, M. S. Hossain, M. Shorfuzzaman, Heuristic optimization of multipulse rectifier for reduced energy consumption, IEEE Transactions on Industrial Informatics 18 (2021) 5515–5526.

[6] A. Jaszcza, Reducing the number of calculations in k-nn by class representatives atb voting (2021).

[7] D. Adeniyi, Z. Wei, Y. Yongquan, Automated web usage data mining and recommendation system using k-nearest neighbor (knn) classification method, Applied Computing and Informatics 12 (2016) 90–108. URL: https://www.sciencedirect.com/science/article/pii/S221083271400026X. doi:https://doi.org/10.1016/j.aci.2014.10.001.

[8] D. Połap, N. Wawrzyniak, M. Włodarczyk-Sielicka, Side-scan sonar analysis using roi analysis and deep neural networks, IEEE Transactions on Geoscience and Remote Sensing 60 (2022) 1–8.

[9] J. Siłka, M. Wieczorek, M. Woźniak, Recurrent neural network model for high-speed train vibration prediction from time series, Neural Computing and Applications 34 (2022) 13305–13318.

[10] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, science 313 (2006) 504–507.

[11] Y. Liao, V. Vemuri, Use of k-nearest neighbor classifier for intrusion detection11an earlier version of this paper is to appear in the proceedings of the 11th usenix security symposium, san francisco, ca, august 2002, Computers Security 21 (2002) 439–448. URL: https://www.sciencedirect.com/science/article/pii/S016740480200514X. doi:https://doi.org/10.1016/S0167-4048(02)00514-X.

[12] G. S. Priya, M. Latha, K. Manoj, S. Prakash, Unusual activity and anomaly detection in surveillance using gmm-knn model, in: 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2021, pp. 1450–1457. doi:10.1109/ICICV50876.2021.9388587.

[13] G. Wang, D. Hoiem, D. Forsyth, Building text features for object image classification, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1367–1374. doi:10.1109/CVPR.2009.5206816.

[14] D. Dua, C. Graff, UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.