SN Computer Science (2023) 4:47
https://doi.org/10.1007/s42979-022-01469-3

ORIGINAL RESEARCH

Optimization Strategies for the *k*-Nearest Neighbor Classifier

Hermann Yepdjio Nkouanga[1] · **Szilárd Vajda**[2]

Received: 15 March 2022 / Accepted: 21 October 2022 / Published online: 10 November 2022

Abstract
In this paper, we propose six (6) fast and efficient classification schemes for different type of images (digits, objects, characters) using the classical *k*-nearest neighbor (*k*NN) classifier. It is common knowledge that *k*NN is one of the most popular supervised classification algorithm. However, for large data collections, the classification is time consuming because of the distance calculations which increase with the number of training samples and the data dimensionality. To reduce the number of training samples, we propose two techniques that employ the notions of convex envelope and stratified sampling. For the convex envelope, only the data points building the convex hull will serve as designated prototypes (training samples), thus reducing considerably the computational burden. To reduce the dimensionality of the data, we considered auto-encoder networks and vector embedding. The former is learning a way to compact data representation using statistical machine learning and the latter is changing the data itself emphasizing how each data sample is organized in space compared to others. The experiments on multiple benchmark data collections such as MNIST, Fashion-MNIST and Lampung characters show a considerable classification speed up (up to 32×) with no significant drop in accuracy when compared to results obtained using the complete data.

Keywords *k*-Nearest neighbor · Convex envelope · Stratified sampling · Autoencoder network · Vector embedding · Digit recognition · Character recognition · Classification

Introduction

Classification is an important goal in machine learning. Given a pattern (an image, an object, a (raw) measurement, etc.) a classification algorithm should be able to determine in which group (class) that pattern belongs. For that reason, current algorithms (except [1, 2]) look for similarities to identify relationships between the patterns.

Mainstream machine learning (ML) uses different feature representations and calculates different distance metrics to distinguish the class labels for the analyzed patterns. Methods such as *k*-Nearest Neighbor (*k*NN), Self Organizing Map (SOM), Growing Neural Gas (GNG), Neural networks (NN), Dynamic Time Warping (DTW), *k*-means clustering, etc. rely on distance calculations to make classification decisions or to back-propagate error in the system in case of neural networks [3]. Using this strategy, one pattern is compared to another using only feature based local comparisons. The distance metrics often used include Euclidean, Hamming, City block, Minkowski, Mahalanobis, Chebychev, Manhattan, Cosine similarity, Jaccard similarity or Spearman coefficient [4].

Though indisputably efficient, the methods that rely on distance calculations are very time costly. The higher the number of data points and/or their dimensionality, the higher the amount of time is needed to calculate the similarities. This is due to the mathematical complexity of distance metrics used in the classification process. If there is no time

Hermann Yepdjio Nkouanga and Szilárd Vajda contributed equally to this work.

This article is part of the topical collection "Advances in Applied Image Processing and Pattern Recognition" guest edited by K C Santosh.

✉ Hermann Yepdjio Nkouanga
hermann@pdx.edu

✉ Szilárd Vajda
szilard.vajda@cwu.edu

[1] Department of Computer Science, Portland State University, 1825 SW Broadway, Portland 97201, OR, USA

[2] Department of Computer Science, Central Washington University, 400 University Way, Ellensburg 98926, WA, USA

constraint, one could calculate all the distances and get the best possible results. However, the run time complexity is primordial in modern real-time applications such as postal automation [5, 6], face matching [7], road sign recognition [8], etc.

To improve the run time complexity of *k*NN classification, we propose several solutions to either reduce the size of the training dataset (class prototypes selection) or to diminish the data complexity (dimensionality) using different feature space transformations. For the former, we reduce the number of samples while keeping the data dimensionality intact. For the latter, we keep all the samples but represent them in a lower dimensional feature space.

For prototypes selection, we propose the concept of convex envelope. However, the detection of the convex hull in large data sets considering high dimensional spaces is bounded by the time limits and memory limits [9]. To avoid such limitations, the data points are reduced to lower dimensions using either Principal Component Analysis (PCA) or Feature Agglomeration (FA). Another technique we used for prototypes selection is called stratified sampling [10]. We considered it due to its success in the field of big data [11].

For the dimensionality reduction, we propose an auto-encoder network able to map a high dimensional space into a low dimensional one by learning the transformations inside the model [12]. We also considered a vector embedding technique which transforms the original feature space into a spatial representation.

The novelty of this paper can be summarized as follows: (i) reduction of the original training set by using elements only from the convex envelope to serve as class prototypes, (ii) solving the convex envelop problem in high dimensional space by reducing the dimensionality of the original data, (iii) the usage of the auto-encoder networks and vector embedding to reduce the data dimensionality and (iv) the combination of the encoder and the vector embedding to produce a new feature representation for the data.

The remainder of the paper is organized as follows. "Related work" discusses the *k*NN paradigm and the different attempts to reduce the time complexity in this classification scheme. In "Method" we introduce the prototype selection based on the convex hull, the encoder network, the stratified sampling and the vector embedding. "Experiments" describes the data collections we used for the experiments and the obtained results. Finally, "Conclusion" summarizes the methods, the results and possible future works.

Related Work

Unequivocally, the *k*-nearest neighbor (*k*NN) can be considered the most used and the most studied supervised classification algorithm. Originally, proposed by Fix and Hodges [13], the *k*NN is a method to classify unknown patterns based on distance calculations to existing, — already identified patterns, by inheriting the label of the pattern which resides the closest considering different distance metrics. Besides classification, the method can also be used for regression.

Formally, the problem can be stated as follows. Let $P = \{p_1, p_2, \dots, p_n\}$ be a template or reference set containing n number of points taking values in R^d, where d denotes the dimension of the data space. Let $Q = \{q_1, q_2, \dots, q_m\}$ be a query set containing m different data points with the same dimension. The k-nearest neighbor problem consists of searching for each point $q_i \in Q$ in the template set P given a specific distance metric.

Despite the rather limited and simplistic non-parametric strategy, *k*NN was used with success in many applications [14, 15]. Torralba et al. [16] argue that if the size of the data is adequate and the object classes to be recognized are rich there is no need for complex classification models. A *k*NN type classifier can perform in the same range as many, nowadays, popular parametric methods.

However, it is common knowledge that the *k*NN is limited by the size of the data, the dimensionality of the data and last but not least the complexity of the distance measure. A large amount of data, an increased data dimensionality and a complex distance measure can lead to tedious and time-consuming calculations. To mitigate this, several solutions focusing on reducing the computational complexity of the *k*NN algorithm have been proposed in the literature over the years.

Some methods use graph-based solutions to approximate the nearest neighbor [17]. Some others use a data subsampling strategy to reduce the number of distance calculations by reducing the number of data points [18–20] considering different selection mechanisms meant to quantify the quality of the selected reference points. In [21] the authors propose a dimensionality reduction on the original data to reduce the number of calculations. In another attempt [22] the authors consider some preliminary clustering followed by a selection process. Garcia et al. [23] focus more on the implementation of the method by utilizing parallel calculation in the GPU.

Recently, an extremely fast implementation of *k*NN called FAISS [24] was released. This implementation is based on building an index in the memory on which the similarity search operates. Despite considering this method in our previous study [25], we currently do not report any results using this method as we solely focus our attention on the *k*NN implementation provided by the Scikit-learn library [24].

To avoid hardware-dependent solutions and those that reduce the quality of the information, our methods are similar to those that do not alter the distance metric and act on the original feature space or carefully selected features. Such

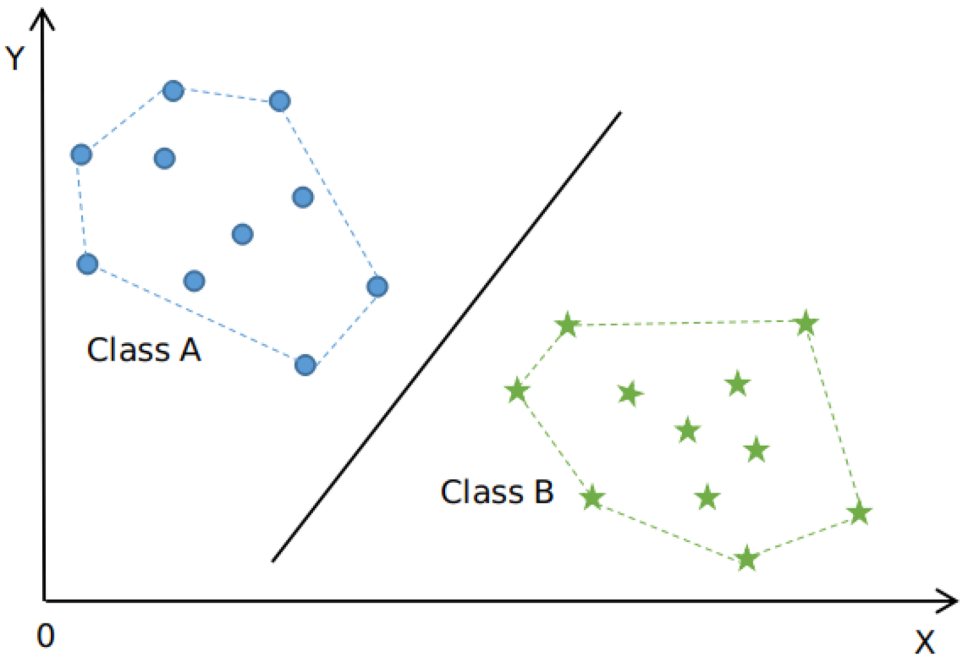

Fig. 1 An intuitive explanation for the convex hull-based classification. The elements on the borders can also represent the elements inside the convex hull

solutions allow keeping the information intact while speeding up the overall *k*NN classification process.

Method

Since we want to utilize the *k*NN as our classifier, the size and dimension of the data are critical. For that reason, instead of using all samples $\forall p_i \in P, i = \overline{1,n}$ as prototypes, the aim is to select only a limited number $p_k \in P, k = \overline{1,k}$ where k is the number ($k \ll n$) of those capable to represent each classes or to select a new feature space of dimension l such as $l \ll d$.

Convex Hull

To select the prototypes we combined two methods. For selection, we would like to use the concept of convex set, more specifically the notion of convex hull. The convex hull or convex envelope is the smallest convex set that contains it. Finding the convex hull of a finite set of 2D points or other low-dimensional Euclidean spaces is a fundamental algorithm in computational geometry.

Usually, for low-dimensional spaces, the complexity of the algorithm is $\mathcal{O}(nlogn)$ and can be obtained by Graham's scan, Chan's algorithm or the Kirkpatrick-Seidel algorithm [9].

An intuitive explanation of our method can be seen in Fig. 1. The elements (data points) constituting the convex hull represent all the points inside the convex envelope, thus we can substitute each class only by those points which build this particular set. A similar concept is applied to the Support Vector Machines (SVM) [26] when the support vectors are identified. In 2D the solution for the convex hull is straightforward, but in higher dimensions, the n-dimensional convex envelope also inherits the same properties. Therefore, the process is limited to calculating the convex set for each class separately.

As mentioned earlier, in high dimensional spaces the convex hull algorithm implementations are bounded by time and memory constraints. To overcome them, we decided to reduce the dimensionality of the data points using either Principal Component Analysis (PCA) or Feature Agglomeration (FA). PCA is a technique that reduces dimensionality by discarding those features that contain the same information. The reduced dimensions are linear combinations of the original variables and they are independent of each other. FA is an unsupervised technique that uses hierarchical clustering to reduce the dimensionality of the data.

Note: We also experimented with an encoder network to reduce the dimensionality of the data. Although such reduction was used with success in comparable scenarios [27], though admittedly using way more dimensions, the data was not separable in such low dimensions. Therefore, we were not able to identify the convex hulls for the different classes. The representation of the MNIST digits in a 2D space, using an encoder network can be observed in Fig. 2. It is to be observed the considerable overlap among the different classes.

Once the convex envelope for each class is defined, those samples that are part of the envelope are identified and designated as the new class prototypes. However, the

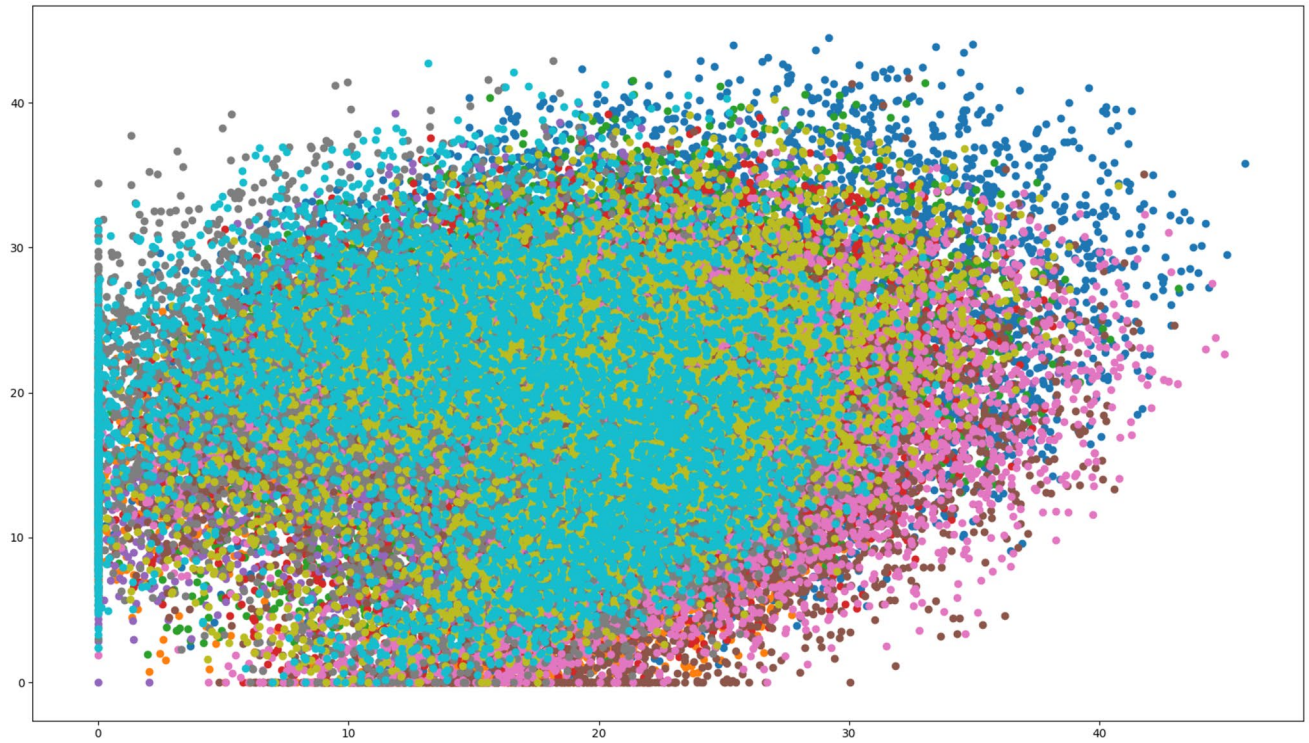

Fig. 2 A 2D representation of the MNIST test collection considering an encoder network to reduce the 784 features to 2. The different colors represent the different digit classes

classification is not happening in this reduced space but in the original space of the data. This mapping to the original space is useful as in this space the data representation is complete. Using the reduced dimensionality space (we set it to six (6) in our experiments based on trial runs) for the classification would significantly impact the results because of the important information loss.

Dissimilarity space embedding

Another way to transform the original feature space is the dissimilarity space embedding [2]. This generic scheme allows the transformation of all kinds of feature spaces into a new representation that emphasizes not the data point itself, but its spatial relationship to other data points by introducing the diss$(\cdot,\cdot)$ metric. The dissimilarity metric (diss(x, y)) can be perceived as a distance measure between the data point in question (x) and the prototype (y). The idea behind this representation is completely based on the dissimilarity measure diss.

Intuitively, the idea of the dissimilarity is to transform an original representation into an n-dimensional vector, where each element in the vector is a distance of the sample in question to some prototype P_i. The new representation for a certain point x will look like $V_x = (\text{diss}(x, p_1), \text{diss}(x, p_2), \ldots, \text{diss}(x, p_n))$. The p_1, p_2, ..., p_n are the prototypes (class representatives) and n controls the number of the prototypes and implicitly the dimensionality of the new feature space.

This dissimilarity space embedding will allow us to measure,- –instead of dealing with raw features, how each data point defines itself compared to its neighbors. However, despite the fact that this transformation is very efficient, the question is how those prototypes (see p_1, p_2, ..., p_n) should be selected. For our experiments, we considered a simple selection technique, namely the k-means clustering. Even though the method is not producing the best results, it has several advantages. First, it is an unsupervised technique, therefore it can be applied without any apriori knowledge or human involvement. Second, the parameter k (see the number of clusters) allows us to have full control over the dimensionality. For our purpose, we considered for each class the same amount of clusters, and the prototypes for each class (see $p_i, i = \overline{1,k}$) were assigned as the class centroids of the clusters.

Other Strategies

To reduce the dimensionality, we involved an auto-encoder type of network [12]. Such a neural network instead of applying some mathematically defined formula to transform the data learns the transformation itself in an unsupervised

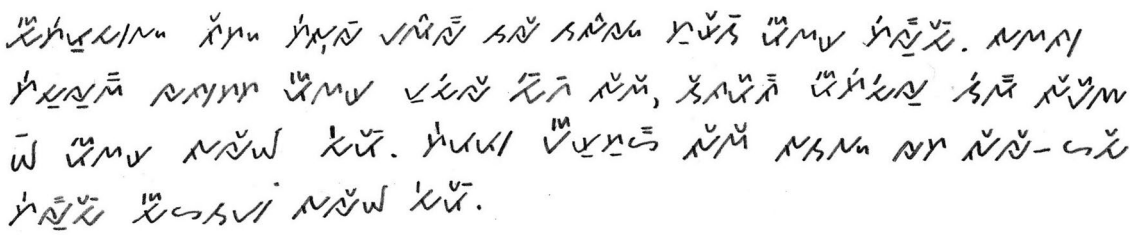

Fig. 3 A text written in Lampung script

manner. This efficient encoding process is governed by optimizing the reconstruction performances of the original input at the output. These models are capable of learning a certain encoding for some data collection, —usually involving dimensionality reduction, by training the model to ignore insignificant data.

For data subsampling, —besides the convex hull method, we borrowed from computational statistics the concept of stratified sampling which can reduce the variance when a Monte Carlo method is used to estimate statistics from a certain number of data points. The advantage of this method is that we can fully control the amount of data the method can retain.

Experiments

First, we present briefly the datasets used in these experiments followed by the achieved results. To be able to judge the quality of the methods, we report our accuracy and speed performance compared to the classical *k*-nearest neighbor utilizing brute force, –which compares each query example with all the reference samples [23].

Data Collections

MNIST [28] is a well-known benchmark dataset[1] containing separated digits assigned to 10 different classes. The images, —coming mainly from US census forms, are size normalized and centered to 28 ×28 gray-level images. The data set contains 60, 000 and 10, 000 images for training and testing, respectively.

Fashion-MNIST [29] is a newly introduced benchmark dataset[2] comprising ten different types of fashion products. The original images were converted to 28x28 gray-scale images. The size of the training and the test set is similar to MNIST. 60, 000 images are considered for training and 10, 000 images are used for testing.

The Lampung characters used in these experiments were extracted from a multi-writer handwritten collection produced by 82 high school students from Bandar Lampung, Indonesia. The Lampung texts are created as transcriptions of some fairy tales. One exemplary document snippet can be seen in Fig. 3. 23, 447 and 7853 characters were collected for training and testing respectively. Altogether, 18 different character classes were identified. Each character is represented by a centered and normalized 32 ×32 gray-scale image. More details about this publicly available data are to be found in [30, 31]. Is to be noted, that for convenience purposes the original images were resized to 28 ×28 keeping the aspect ratio intact.

Experimental System

All our experiments were performed on an Intel 2.8 GHz machine equipped with 24 GB of RAM running Ubuntu 18.04. For the *k*NN classification experiments, we considered the *k*NN algorithm implemented in the Scikit-learn library [32].

For the encoder network, we considered the solution proposed by Zhang [33]. Some optimization strategies such as manual search, random search, grid search, etc. implemented in frameworks such as Optuna [34] or HyperOpt [35] could have provided a more compact encoding. However, such parameter optimization was currently out of the scope for these experiments.

Results

In this section, we will present several results. We aim to show how the methods we propose perform on different data benchmarks. For all our experiments we considered accuracy as a measurement of success. We also recorded the elapsed time for each experiment to measure the impact of each method on the speed component.

To compare the performances of our strategies, we defined a golden standard, namely the accuracy and the elapsed time when the *k*NN [32] is applied to the complete dataset. This will be considered as the baseline system for all our experiments. For all our trials, we considered the L_2

[1] http://yann.lecun.com/exdb/mnist/.

[2] https://github.com/zalandoresearch/fashion-mnist.

Table 1 *k*NN [32] classification results for MNIST

Method	# Samples/dimension	Acc. (%)	Time (s)
Baseline	60,000 (784)	96.91	1873
PCA	19,943 (784)	94.91	270
FA	13,555 (784)	93.97	167
Strat. samp.	54,000 (784)	96.73	830
Encoder	60,000 (50)	97.7	57
Feat. embed.	60,000 (250)	95.14	48
Combination	60,000 (100)	97.63	79

Table 2 *k*NN [32] classification results for Fashion-MNIST

Method	# Samples/dimension	Acc. (%)	Time (s)
Baseline	60,000 (784)	84.97	684
PCA	10,421 (784)	74.76	124
FA	9077 (784)	73.01	115
Strat. samp.	54,000 (784)	84.62	615
Encoder	60,000 (150)	86.88	73
Feat. embed.	60,000 (300)	77.79	20
Combination	60,000 (100)	86.04	31

Table 3 *k*NN [32] classification results for Lampung

Method	# Samples/dimension	Acc. (%)	Time (s)
Baseline	23,447 (784)	83.94	306
PCA	8848 (784)	73.97	132
FA	8022 (784)	76.49	115
Strat. samp.	–	–	–
Encoder	23,447 (100)	90.92	25
Feat. embed.	23,447 (540)	80.48	48
Combination	23,447 (140)	89.63	27

(Euclidean) norm as the distance metric for the *k*NN and *k* (size of the neighborhood) was set to 1.

In Table 1 we can observe that the baseline method, –using all 60.000 MNIST digit samples from the training dataset reaches a high recognition score of 96.91%. To achieve this, the *k*NN implementation from [32] spent approximately 1873 s to calculate Euclidean distances for all the test samples. Considering that we have 60,000 train samples and 10,000 test samples, altogether 600,000,000 distance calculations were necessary which indisputably take a considerable amount of time.

However, once we apply the reduction strategies to reduce the number of reference points, the time is reduced to 270 and 167 s for PCA and FA, respectively. A more descending trend is to be observed for the vector embedding (see 48 s), the encoder (see 57 s), and for the combination of the two methods (see 79 s). It is to be noted, that not only the execution time diminishes considerably, but also the accuracy increases for the encoder and combination method.

For the encoder solution, —when 784 features were encoded into 50 features by optimizing the reconstruction error, the accuracy gain is 0.79%, while the speed gain is 32.8×. Similarly, for the combination (50 features coming from the encoder network and the remaining 50 from vector embedding) the accuracy gain is 0.72%, while the speed gain is 23.7×. This leads us to the conclusion that it is possible to achieve more accuracy with less computation burden. For the stratified sampling and the vector embedding, the speed gain is considerable but the accuracy is decreasing a bit compared to the baseline method.

The Fashion-MNIST results are shown in Table 2. The baseline system reports 84.97% accuracy when all 60.000 items were considered as training material. With the reduction (see PCA) the accuracy loss is 10.21% but the speed gain is 5.5×. However, similarly to the previous case (see MNIST results in Table 1), the encoder and the combination method outperform the baseline system. The encoder shows a net accuracy gain of 1.92% with a speed acceleration of 9.3×, while the combination method has a net gain of 1.07% and an acceleration of 22×. Admittedly, the accuracy gain for the combination method is less than in the case of the encoder, but the speed gain is more than double, which in real-time applications could be a huge advantage.

Similar trend observed in Table 2 can also be observed in Table 3 when the data reduction strategy is applied to Lampung characters. The data is reduced by 65%, the accuracy loss is 7.45% and the speed gain almost tripled for the PCA solution combined with the convex envelop. However, the results reported for the encoder solution and the combination method are astonishing. For the encoder the accuracy gain is 6.98% and the speed-up factor is 12.2×. For the combination the accuracy gain is 5.69% and the speed-up factor is 11.3×.

Note: One could observe that there are no results reported for the stratified sampling for the Lampung data (see more in Table 3). This is due to the fact that during the clustering which precedes the stratified sampling some clusters contain only one sample which prohibits the stratified sampling to select the prototypes.

Thoroughly analyzing all the results, one could observe the following trends. For each collection, the encoder solution provides the best results and usually the highest speed gain (except for the Lampung collection), which leads to the conclusion that all these collections contain redundant information and that can lead to some confusion. The autoencoder networks are very reliable tools to diminish data dimensionality and due to the strict optimization rule governed only by the reconstruction error minimization, the representation is very compact and meaningful. Another advantage of this method is its adaptability as the size of the bottleneck is adjustable.

An interesting conclusion is related to the vector space embedding. The technique is very useful, and it can transform all kinds of feature representation into a spatial representation where the original features do not contribute much but rather how each data point is positioned compared to other data points. In our experiments, we have seen that for this method the speed gain is considerable even though the classification accuracy falls short behind the baseline method. However, when the method is paired with the encoder-generated features, the results considerably outperform the baseline method and also the convex hull-related strategies. This leads us to the conclusion that the encoder-generated features and the spatial distribution of the data points complement each other forming a very strong bond that can be considered as an alternative feature representation.

When looking at the results from the three datasets, we can see that all the solutions (including the baseline one) perform significantly better for MNIST than for the other datasets in terms of classification accuracy. For example, the baseline system yields 96.91% for MNIST but only 84.97% and 83.94% for Fashion-MNIST and Lampung, respectively. The encoder system yields 97.7% for MNIST but only 86.88% and 90.92% for Fashion-MNIST and Lampung, respectively. In terms of classification time, the gain for all the solutions (across all the datasets) compared to their corresponding baseline systems is proportional to the amount of data that is reduced from the original data.

Despite the fact that all three data collections are different: (a) one is a large collection of rather simple digits (see MNIST), (b) the other is similar in size but represents fashion items (see Fashion-MNIST) and (c) last the Lampung characters collection is less numerous but representing almost twice as many classes and more complex in the shapes, one could say that the trend of these methods is similar. All of them substantially reduce the execution time for the classification. For all the collections, some of the proposed methods obtain even better scores than if we would have either considered the complete collection or kept the original feature representations. This shows the efficiency of the methods in terms of speed and accuracy.

Before reaching to the conclusion, one should discuss the limitations of the convex hull-based solutions. While the gain is valuable for the MNIST, the accuracy drop for the other two collections might not make these methods very attractive. The current results lead to the conclusion that for these collections the convexity notion might not be straightforward. The different class-wise convex hulls can overlap and create confusion between different classes. Due to the computational complexity of the convex hull, the reduction of the feature space (see PCA or FA) will also hardly influence the exact calculation of the convex envelopes.

Conclusion

In this manuscript, we proposed several efficient and easy-to-implement solutions to reduce the complexity of the *k*-nearest neighbor classifier. This classical strategy is very efficient and easy to apply to all kinds of classification tasks. However, due to the tremendous distance calculations between the query item and the prototype items, this classifier should be avoided when there is a time constraint such as in real-time applications. The classification time will grow considerably once we have many training samples (see prototypes) and the dimensionality of the data is high.

For that reason, we concentrated our effort to develop different strategies to mitigate this time constraint. The proposed solutions focus on either reducing the number of prototype samples (i.e. FA, PCA and stratified sampling) or transforming the feature space to a more compact representation of the same data (i.e. encoder, vector embedding and their combination).

The first strategy is trying to discard prototypes that do not contribute to the classification, thus creating an excess of calculations. The second strategy is keeping the prototypes in place but focusing on reducing the dimensionality of the data. Both strategies try to achieve the same outcome: reduce the computational burden.

First, to reduce the number of reference points we considered only the data points which build the convex hull. All the points inside the convex envelope were discarded. However, due to the time and memory complexity of identifying the convex envelope, a data reduction strategy was necessary. We experimented with PCA and FA, respectively. The high dimension (see 784) of each data collection was reduced to six (6), a condition which ensured to run the convex hull algorithm with success. However, as it can be observed the results are somehow ambiguous. There is a substantial speed gain, however, the accuracy starts to drop considerably for the MNIST-Fashion and Lampung. Second, the stratified sampling is reporting way more promising results by considerably increasing the speed component while staying close right behind the baseline system (which uses all the prototype samples) in terms of accuracy.

For the feature space transformation, the encoder network, the vector space embedding and their combination outperform all the previous approaches. The encoder network transformed all feature representations into more compact ones, where besides considerable speed gains (see MNIST: 32.8× , MNIST-Fashion: 9.3× and Lampung: 12.2×) the performances did not drop, but rather improved by 0.72%, 1.92%, and 6.98% for MNIST, MNIST Fashion, and Lampung respectively. While the individual results for

the feature embedding are fading behind the encoded version, the combination of the two is rather impressive. This suggests that these features complement each other and the spatial disposition of the data points relative to other data points is an important characteristic.

To further improve the convex hull-related methods, one could think of calculating the convex envelope differently. Instead of calculating the envelope using all the elements, some unsupervised clustering could be applied and several smaller convex hulls could be established. Later, these convex hulls could be merged into a global one. For the encoder network, some more sophisticated architecture could yield even higher performances.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Duin RPW, Verzakov S. Fast knn mode seeking clustering applied to active learning. CoRR abs/1712.07454 (2017).
2. Pekalska E, Duin RPW, Paclík P. Prototype selection for dissimilarity-based classifiers. Pattern Recognit. 2006;39(2):189–208.
3. Bishop CM. Neural networks for pattern recognition. New York: Oxford University Press Inc; 1995.
4. Irani J, Pise N, Phatak M. Clustering techniques and the similarity measures used in clustering: a survey. Int J Comput Appl. 2016;134:9–14.
5. Sharma N, Sengupta A, Sharma R, Pal U, Blumenstein M. Pincode detection using deep CNN for postal automation. In: International conference on image and vision computing, 2017;1–6.
6. Vajda S, Roy K, Pal U, Chaudhuri BB, Belaïd A. Automation of Indian postal documents written in Bangla and English. Int J Pattern Recognit Artif Intell. 2009;23(8):1599–632.
7. Borovikov E, Vajda S. Facematch: real-world face image retrieval. In: Santosh, K.C., Hangarge, M., Bevilacqua, V., Negi, A. editors. Recent trends in image processing and pattern recognition—First international conference, RTIP2R 2016, Bidar, India, December 16–17, 2016, Revised selected papers. Communications in computer and information science, 2016; 709, pp. 405–419.
8. Wang C. Research and application of traffic sign detection and recognition based on deep learning. In: 2018 international conference on robots intelligent system (ICRIS), 2018;150–152.
9. Avis D, Bremner D. How good are convex hull algorithms? In: Snoeyink, J. editor. Proceedings of the eleventh annual symposium on computational geometry, Vancouver, B.C., Canada, June 5–12, 1995, pp. 20–28.
10. Shahrokh Esfahani M, Dougherty ER. Effect of separate sampling on classification accuracy. Bioinformatics, 2013;30(2), 242–250. https://doi.org/10.1093/bioinformatics/btt662. https://academic.oup.com/bioinformatics/article-pdf/30/2/242/17147301/btt662.pdf.
11. Liu Z, Zhang A. A survey on sampling and profiling over big data (technical report). CoRR abs/2005.05079. 2020.
12. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006;313(5786):504–7.
13. Fix E, Hodges JL. Discriminatory analysis: nonparametric discrimination, consistency properties. 1951.
14. Agarwal Y, Poornalatha G. Analysis of the nearest neighbor classifiers: a review. In: Chiplunkar, N., Fukao, T. editors. Advances in artificial intelligence and data engineering—Select proceedings of AIDE 2019. Advances in intelligent systems and computing, pp. 559–570. Springer, Berlin, 2021. International conference on artificial intelligence and data engineering, AIDE 2019; Conference date: 23-05-2019 through 24-05-2019.
15. Taunk K, De S, Verma S, Swetapadma A. A brief review of nearest neighbor algorithm for learning and classification. In: 2019 international conference on intelligent computing and control systems (ICCS), pp. 1255–1260, 2019.
16. Torralba A, Fergus R, Freeman WT. 80 million tiny images: a large data set for nonparametric object and scene recognition. PAMI. 2008;30(11):1958–70.
17. Hajebi K, Abbasi-Yadkori Y, Shahbazi H, Zhang H. Fast approximate nearest-neighbor search with k-nearest neighbor graph. In: Proceedings of the twenty-second international joint conference on artificial intelligence-vol. 2. 2011;1312–1317.
18. Bentley JL. Multidimensional divide-and-conquer. Commun ACM. 1980;23(4):214–29.
19. Friedman JH, Bentley JL, Finkel RA. An algorithm for finding best matches in logarithmic expected time. ACM Trans Math Softw. 1977;3(3):209–26.
20. Indyk P, Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on theory of computing. STOC '98, pp. 604–613. ACM, New York; 1998.
21. Jalan A, Kar P. Accelerating extreme classification via adaptive feature agglomeration. CoRR abs/1905.11769. arXiv:1905.11769. 2019.
22. Vajda S, Santosh KC. A fast k-nearest neighbor classifier using unsupervised clustering. In: Santosh KC, Hangarge M, Bevilacqua V, Negi A editors. Recent trends in image processing and pattern recognition—first international conference, RTIP2R 2016, Bidar, India, December 16–17, 2016, Revised selected papers. Communications in computer and information science, vol. 709, pp. 185–193, 2016.
23. Garcia V, Debreuve E, Barlaud M. Fast k nearest neighbor search using GPU. In: 2008 IEEE Computer Society conference on computer vision and pattern recognition workshops, pp. 1–6, 2008.
24. Johnson J, Douze M, Jégou H. Billion-scale similarity search with gpus. CoRR abs/1702.08734. 2017. arXiv:1702.08734.
25. Yepdjio H, Vajda S. A fast and efficient k-nearest neighbor classifier using a convex envelop. In: International conference on recent trends in image processing and pattern recognition, RTIP2R 2021, Msida, Malta, December 8–10. Communications in Computer and Information Science, 2021.
26. Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995;20(3):273–97.
27. Vajda S, Rangoni Y, Cecotti H. Semi-automatic ground truth generation using unsupervised clustering and limited manual labeling: application to handwritten character recognition. Pattern Recognit Lett. 2015;58:23–8.
28. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. In: Intelligent signal processing, pp. 306–351, 2001.
29. Xiao H, Rasul K, Vollgraf R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR abs/1708.07747; 2017.
30. Junaidi A, Vajda S, Fink GA. Lampung—a new handwritten character benchmark: database, labeling and recognition. In: International workshop on multilingual OCR (MOCR), pp. 105–112. ACM, Beijing, 2011.

31. Vajda S, Junaidi A, Fink GA. A semi-supervised ensemble learning approach for character labeling with minimal human effort. In: ICDAR, pp. 259–263, 2011.
32. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
33. Computer Science University of Toronto: transpose convolutions and autoencoders. https://www.cs.toronto.edu/~lczhang/360/lec/w05/autoencoder.html. Accessed 03 Jan 2021.
34. Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD international conference on knowledge discovery and data mining. 2019.
35. Bergstra J, Komer B, Eliasmith C, Yamins D, Cox DD. Hyperopt: a python library for model selection and hyperparameter optimization. Comput Sci Discov. 2015;8(1): 014008.