Clustering-based k-Nearest Neighbor Classification for Large-Scale Data with Neural Codes Representation

Antonio-Javier Gallego, Jorge Calvo-Zaragoza*, Jose J. Valero-Mas, Juan Ramón Rico-Juan

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Carretera San Vicente del Raspeig s/n, Alicante, 03690, Spain

Abstract

While standing as one of the most widely considered and successful supervised classification algorithms, the k-Nearest Neighbor (kNN) classifier generally depicts a poor efficiency due to being an instance-based method. In this sense, Approximated Similarity Search (ASS) stands as a possible alternative to improve those efficiency issues at the expense of typically lowering the performance of the classifier. In this paper we take as initial point an ASS strategy based on clustering. We then improve its performance by solving issues related to instances located close to the cluster boundaries by enlarging their size and considering the use of Deep Neural Networks for learning a suitable representation for the classification task at issue. Results using a collection of eight different datasets show that the combined use of these two strategies entails a significant improvement in the accuracy performance, with a considerable reduction in the number of distances needed to classify a sample in comparison to the basic kNN rule.

Keywords: Efficient kNN classification, Clustering, Deep Neural Networks

*Corresponding author: Tel.: +349-65-903772; Fax: +349-65-909326

Email addresses: jgallego@dlsi.ua.es (Antonio-Javier Gallego), jcalvo@dlsi.ua.es (Jorge Calvo-Zaragoza), jjvalero@dlsi.ua.es (Jose J. Valero-Mas), juanramonrico@ua.es (Juan Ramón Rico-Juan)

1. Introduction

The k-Nearest Neighbor (kNN) classifier represents one of the most widely used schemes for supervised learning tasks [6]. This method only requires that a dissimilarity can be defined between two given instances. Basically, kNN classifies a given input element by assigning the most common label among its k-nearest prototypes of the training set according to that dissimilarity.

Most of its popularity comes from its conceptual simplicity and straightforward implementation, that deals surprisingly well in many pattern recognition tasks. In addition, it is well suited to problems facing *multi-class classifications*, that is, those in which the set of possible labels contains more than two elements [7]. In this sense, unlike other algorithms such as Support Vector Machines, which have to choose some kind of strategy to adapt to this scenario [8], the kNN rule does not have to make any adjustment since it is naturally multi-class.

As a representative example of instance-based algorithm, the kNN classifier does not perform an explicit generalization process (i.e., building a model) out of the initial training data but directly considers those samples for classification [9]. The classifier therefore improves its performance as the training set increases, having been demonstrated its consistency as the number of training instances approaches to infinity [10].

Fortunately, our society is strongly characterized by the large amount of information surrounding us. Since the start of the information-related technologies, data production has been reported as constantly growing [11], being this effect more remarkable in recent years. Therefore, a kNN classifier may be able to exploit these large-scale sources of information to improve classification performance.

Nevertheless, since the kNN classifier needs to compute a distance between the input sample and every single sample of the training data, it entails low efficiency in both classification time and memory usage. This constitutes the main drawback for this classifier, which becomes an insurmountable obstacle

when considering such large-scale training corpora.

In this work we use an efficient search based on a clustering strategy. The main assumption is that the k-nearest neighbors of a given instance lie in the same cluster. Thus, the kNN search can be efficiently performed in two steps: i) reaching the nearest cluster; and ii) finding the k-nearest neighbors within the cluster. In a large-scale scenario, this would eventually save a huge amount of distance computations, thereby performing the process more efficiently [12]. Yet there is a possibility that this search entails some accuracy loss if part of the k-nearest neighbors fall in different clusters. To alleviate this situation, we consider a strategy so that this possibility is more unlikely. Our idea is that clusters are not necessarily disjoint but there are instances that can belong to more than one. For achieving that, we apply an additional step after the initial clustering process: (i) focusing on one cluster, we iterate through each of the instances; (ii) for each element of the cluster we check the k-nearest neighbors considering the entire training set; (iii) in case any of the k neighbors of the instance at issue is not part of the cluster being examined, we include it inside the cluster, thus approaching the space partitioning to something similar to a fuzzy clustering; (iv) this process is done for each of the clusters obtained. This strategy increases the likelihood of making all the k-nearest neighbors of a given test instance fall in the same cluster. Also note that both the clustering process and the proposed enlargement are done as a preprocessing stage, thus not affecting the efficiency of the classification process. As it shall be later experimentally checked, this process of increasing the cluster size approaches the brute-force kNN scenario in terms of accuracy with far less computational cost.

Furthermore, recent advances in feature learning, namely deep learning, have made a breakthrough in the ability to learn suitable features for classification. That is, instead of resorting to hand-crafted features extracted, the models are trained to infer out of the raw input signal the most suitable features for the task at hand. This representational learning is performed by means of Deep Neural Networks (DNN), consisting of a number of layers which are able to

represent different levels of abstraction out of the input data. Some authors [13, 14], however, have shown that it is interesting to use these DNNs only as feature extractor engines, that is, feeding the network with the input data and taking one of the intermediate representations, most typically the second-to-last layer output, as features for the classification task.

The kNN method may obtain more complex decision boundaries than the common *softmax* activation used in the last layer of a DNN. However, it requires the features and/or the distance considered to be adequate for the task. Taking into account that DNN and kNN are totally complementary in terms of feature extraction and decision boundaries, it is interesting to propose a hybrid system in which both algorithms can exploit their potential and see their drawbacks mitigated by the other. The goodness of a hybrid approach has been demonstrated with other classifiers such as Support Vector Machines [15], yet a comprehensive study in the case of large-scale data with kNN remains open. Note that defining input data with the most appropriate features might have a considerable impact on the performance of the clustering algorithm. Therefore, it is to be expected that these suitable features will also help to improve the approximate search of the kNN.

For all the above, this document presents the following contributions:

1. A new scheme to conduct cluster-based kNN search. We also extend this search with overlapped clusters, and demonstrate that this extension is able to achieve better classification rates than the regular one without significantly increasing the number of distances to compute.
2. The use of DNN for extracting meaningful features as a general framework to improve both accuracy and efficiency of the proposed cluster-based kNN search.
3. A comprehensive experimentation on the issues described above, including several scenarios and heterogeneous datasets, with an in-depth analysis of the reported results supported by statistical significance tests.

The rest of the paper is organized as follows: related background to the topic

of the paper is introduced in Section 2; our proposed approach is developed thoroughly in Section 3; Section 4 describes the experimental set-up considered; the results obtained as well as their analysis are introduced in Section 5; finally, general conclusions obtained from the work are discussed in Section 6.

2. Background

2.1. Efficiency of the k-Nearest Neighbor rule

As a representative example of lazy learning, the kNN classification rule generally exhibits a very poor efficiency: since no model is built from the training data, all training information has to be consulted each time a new element is classified. This fact has two clear implications: on the one hand, high storage requirements; on the other hand, an elevated computational cost. Some variants of the kNN include a training process, such as the work of Zhang et al. [3], in which a model is build to infer the optimal k for each sample. However, it does not reduce the cost when predicting a sample.

These shortcomings have been widely analyzed in the literature and several strategies have been proposed to tackle them. In general, they can be divided into three categories: Fast Similarity Search (FSS) [16], Data Reduction (DR) [17], and Approximated Similarity Search (ASS) [18].

FSS is a family of methods that bases its performance on the creation of search models for fast prototype retrieval in the training set. Generally, these strategies are further subdivided into indexing algorithms [19] and AESA family [20]. The former family represents the set of algorithms which iteratively partition the search space and build tree structures for an efficient search; for a new element to be classified, the search throughout the tree selects the proper space partition (leaf node in the tree) for then performing an exhaustive search within the prototypes in that region; this implies that only a subset of the total number of examples has to be queried for classifying a new instance. Some examples of these methods and structures are *k-d tree* [19], *ball tree* [21], and *metric-trees* [22], among others. The problem, however, is that they are ex-

tremely sensitive to the curse of dimensionality. Also, they require that input data is represented as feature vectors. AESA algorithms, on the other hand, only need a metric space, i.e. that in which a pairwise distance can be defined. These strategies make use of pre-computed distances and the triangle inequality to discard prototypes. The main disadvantage of these algorithms is that only deal with searches involving $k = 1$ and become inefficient with large-scale data. In addition to these techniques, there are also studies that considered specific computing engines like Apache Spark [1] to perform this search efficiently [5, 4].

DR comprises a subset of the Data Preprocessing strategies that aim at reducing the size of the initial training set while keeping the same recognition performance [17]. The two most common approaches are Prototype Generation and Prototype Selection [23]. The former creates new artificial data to replace the initial set while the latter simply selects certain elements from that set. The *Condensed Nearest Neighbor* [24] was one of the first techniques developed for this purpose, yet a number of proposals can be found in the literature under both selection [25] and generation [26] paradigms. More recently, there have been a number of new proposals such as Instance Reduction Algorithm using Hyperrectangle Clustering [27], Reduction through Homogeneous Clusters [1], or Edited Natural Neighbor [2]. The main problem with these methods is that they generally carry a significant loss of accuracy in the classification [17]. Thus, different strategies have been proposed for solving those deficiencies as, for instance, considering boosting schemes [28], merging feature and prototype selection by means of genetic algorithms [29, 26], or considering the results of these reduction algorithms as a means of constraining the space of prototypes to assess by the classifier, namely kNNc [30].

ASS approaches work on the premise of searching sufficiently similar prototypes to a given query in the training set, instead of retrieving the exact nearest instance, at the cost of slightly decreasing the classification accuracy. When large datasets are present in a Pattern Recognition task, the ASS framework

[1] https://spark.apache.org/

rises as a suitable option to consider since possible drawbacks as, for instance, accuracy loss because of not retrieving the actual nearest prototype, are mitigated by the huge amount of information available. Some particular successful principles within this family are the use of hashing techniques to codify the prototypes of the training set. Typical examples comprise the Local Sensitive Hashing (LSH) forest [31], Spectral Hashing [32] or Product Quantization [33]. A different approach is the use of approximate k-d trees for the search (e.g., the Fast Library for Approximate Nearest Neighbors [34]).

Within the context of improving the efficiency of the nearest neighbor search, we also propose an approximate search based on the use of clusters.

2.2. Neural Codes representation

Deep Neural Networks (DNN) are multi-layer architectures designed to extract high-level representations of a given input. They have drastically improved the state-of-the-art in a number of fields and applications such as image, video, speech and audio recognition tasks [35]. Due to their high generalization power, transfer learning can be used to apply DNN models trained on a domain to a different task where data are similar but the classes are different [36]. This transfer can be done by fine-tuning the weights of the pre-trained network on the new dataset by continuing the back-propagation [37] or, alternatively, it can also be performed by using the DNN as a fixed feature extractor to obtain a mid-level representation, forwarding samples through the network to get the activations from one of the last hidden layers, usually a pooling one.

Although extracting such deep representations, referred to as Neural Codes (NC) [38], and then apply kNN search is a common transfer learning technique, to our knowledge there are few comprehensive studies of the kNN classifier outperforming the network in the same domain in which it was trained as, for instance, the work by Ren et al. [39]. In addition, representing input data appropriately not only affects kNN performance, which relies almost exclusively on data representation, but also might have a big impact on the approximate search algorithm. As this fact is especially relevant in clustering approaches —

which is another process strongly dependent on data representation — in this work we study the effect of a NC representation in both the effectiveness and efficiency of the proposed approximate search of the kNN classifier.

3. Clustering-based k-Nearest Neighbor classification with Neural Codes representation

This section presents the proposed scheme to apply kNN effectively and efficiently by means of NC representations and a space partition based on clustering.

The justification of our scheme can be explained by the following terms. The last layer of a neural network used for classification only learns a linear function. That is, it is necessary that the data is presented to this layer in a way that is linearly separable. Therefore, the rest of the layers of a deep neural network behave mainly as a feature extractor, which maps the input to a space in which categories are expected to be linearly separable.

This provides a number of advantages over other procedures. The idea of using hand-crafted features can be useful in a given context, but has to be performed for each possible task independently. In addition, features tend to be meaningful for humans, which is not necessary appropriate for machine learning. Furthermore, a kernel function, such as those typically used in Support Vector Machines, also maps data onto a linearly separable space, but these functions must also be chosen from a limited set of options. The idea, therefore, of deep neural networks is that they provide a rather general approach to learn this mapping, which does not require a priori knowledge of the problem.

Thus, these NC are not only useful in the context of a last neural layer but actually allow a good representation for the problem. This is especially interesting in the case of clustering-based classification since the distribution of the samples in the space is the key aspect in the composition of the clusters. It will be proved during the experimentation that the use of NC obtained by deep networks not only achieves a higher classification accuracy, but also favors

efficiency by producing more suitable clusters for classification.

Below we thoroughly describe the two involved stages: the extraction of NC representation and the data clustering.

3.1. Extraction of Neural Codes representation

Although there are several ways to use deep models for unsupervised learning (such as auto-encoders), we focus on using deep neural networks to learn the aforementioned NC, that is, a feature-based representation of the input data directly derived from the network. Thus, the first step in our process is to train the network in a supervised fashion by providing pairs that contain the element itself (input) and its label.

Note that the intrinsic characteristics of deep neural networks make them especially suitable for the problem at issue. Generally, these models derive similar feature-based representations for different instances of the same class. In principle, this fact supposes an additional advantage in terms of performance when applying a clustering-based search process as most instances representing the same class shall be gathered in a single partition rather than being spread among several of them.

As will be seen below, several heterogeneous datasets will be used to validate the goodness of our proposal. Therefore, a network model has been tuned for each of dataset. This tuning process is aim at obtaining a network that reports competitive results with respect to the state of the art. More details about these models are facilitated in Section 4.

3.2. Space partitioning with clustering

An efficient, yet approximate kNN can be straightforward achieved by using clusters. Let c be the number of clusters chosen, a c-clustering process is performed so that data is grouped into c different partitions trying to minimize some specific criteria (which depend on the particular clustering strategy). Once this process has been performed, the k-nearest neighbor search for a given

sample x consists of, first, retrieving the nearest cluster (represented by its centroid). Subsequently, the conventional kNN search is performed but restricted to those elements that belong to that cluster.

If m denotes the size of the training set, the expected or theoretical number of distances for classifying an input can be estimated as:

$$f(m, c) = c + \frac{m}{c}$$

The first step is to retrieve the nearest cluster, and so c distances are needed. Then, since all the samples must belong to a cluster, an average of $\frac{m}{c}$ distances are needed to search within the retrieved cluster in the next step. The actual number of distances for a given test sample, however, depends on both the cluster retrieved in the first step and the distribution of the samples within the different clusters.

Note that the expected number of distances does not always decreases as the number of clusters increases but there is a point in which the number of distances to each cluster centroid is higher than the cluster itself. Since m is not a free parameter, the optimum number of clusters to minimize the expected number of distances is $\sqrt{m}$ (for which $2\sqrt{m}$ distances are needed, on average). Note, however, that fixing c to this value minimizes the expected number of distance but it does not mean that this configuration leads to the best classification accuracy.

For our work, the clustering process will be performed following the *c-means* algorithm [40], one of the most common and successful algorithms for data clustering [41].[2]

This algorithm follows an *expectation-maximization* approach, in which iteratively samples are assigned to their nearest centroid, and then a set of new centroids are computed to minimize the distance to their samples. In order to provide a more robust clustering, the initialization of the method is performed

[2]This method is also referred to as *k-means*, but we do not want to mislead the meaning of the parameter k of kNN classification.

as described for *c-means++* method [42]. This algorithm proposes an initialization (first centroids) that is expected to provide better results and faster convergence. The algorithm starts with a single random center and the rest of the centers are chosen randomly following a decreasing probability with respect to the distance to the nearest centroid already selected.

Furthermore, as discussed above, these approximated approaches allow to perform the search more efficiently but they usually lead to a loss of accuracy in the classification. Considering the operation of the aforementioned clustering-based search, this loss would be given by those prototypes that, even being in the k-nearest neighborhood of the input query, belong to a different cluster of that selected in the first step.

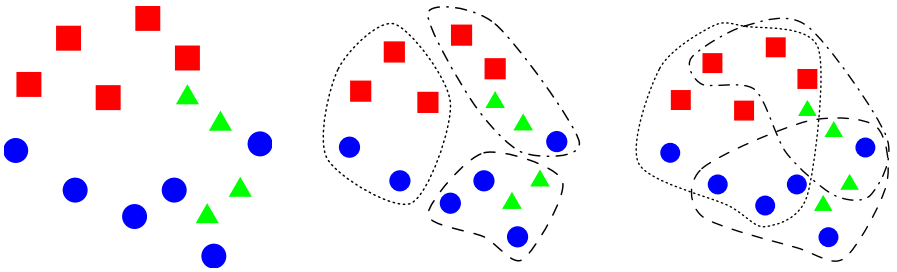
To alleviate this effect, we propose an extension to the cluster-based approach described that consists in slightly increasing the size of the clusters to allow them to overlap. A description of this step is given in Algorithm 1.

Specifically, the c-means++ returns the set C of clusters computed (line 1). Note that $|C| = c$. An iteration is done for every cluster (line 2), after which all its samples are consulted (line 3). Then, all the k-nearest neighbors of these samples are incorporated to that cluster (lines 4-5). Therefore, the probability for the k-nearest neighbors of a test sample to belong to different clusters is reduced.

Algorithm 1 ckNN+ clustering process

Require: $c \in \mathbb{N}, k \in \mathbb{N}; T = (x_i, y_i)_{i=1}^{|T|}$

1: $C \leftarrow$ cMeans++(T, c)
2: **for** $C_i \in C$ **do**
3: **for** $x_j \in C_i$ **do**
4: $N_{x_j} \leftarrow$ kNNsamples(x_j, k, T)
5: $C_i \leftarrow C_i \cup N_{x_j}$
6: **end for**
7: **end for**
8: **return** C



(a) Initial set of instances. (b) Search in the initial cluster case, ckNN. (c) Search in the augmented cluster case, ckNN+.

Figure 1: Visual illustration of the cluster augmentation process.

Throughout the rest of the work we refer to this strategy as *ckNN+* whereas we use the name *ckNN* to the case in which the cluster augmentation process is not applied (avoiding lines from 2 to 7 in Algorithm 1. Figure 1 shows a graphical example of the difference.

It should be noted that, despite adding a measure that aims at improving the accuracy of the classification, this new approach requires a slightly higher number of distances, so finding the best trade-off between efficiency and accuracy must be measured experimentally.

3.3. Classification

The two previous stages (deep network training and creation of extended clusters) can be seen as preprocesses, since only the training set is necessary and can be performed completely before the classification stage. In other words, they do not affect the efficiency of the algorithm in practice.

At the time of classifying unseen samples, our proposal comprises a series of sequential stages (shown at a glance in Fig. 2):

1. The raw sample is propagated through the learned network and its feature vector is extracted from the second-to-last layer (NC representation).
2. The nearest cluster of the sample is calculated and the training instances belonging to it are retrieved.

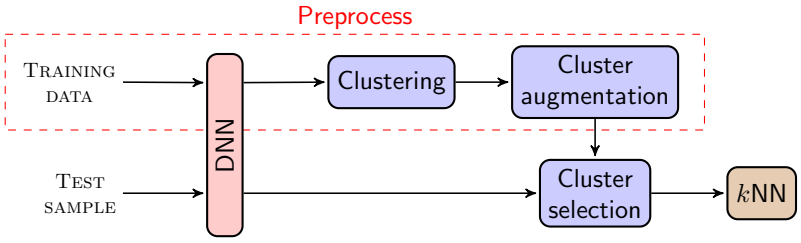


Figure 2: Diagram of the process.

3. A conventional kNN search is performed within these retrieved data.

The idea of the approach is that these processes complement each other perfectly and exploit several synergies for the case of searching in large-scale. On the one hand, such amount of information allows the deep network to learn a good internal representation, while also tending to show the asymptotic guarantees of the kNN. On the other hand, the disadvantages of the approximate search are palliated by both the good representation of the data and the large amount of data, which in addition to the extended version of the clusters may mitigate such disadvantages.

4. Experimental set-up

In this section, we describe the experimental set-up considered for our experiments. This includes the data collection and representation, the neural network topologies, the evaluation methodology, and the related works used for comparative purposes.

All the experiments are performed on Python programming language, using *TensorFlow* (neural network library, version 1.2) and *Scikit-learn* (machine learning library, version 0.18). The machine used consists of an Intel(R) Core(TM) i7-4790K CPU running at 4.00GHz. For high-computing performance, we used a GeForce GTX 980 GPU with the cuDNN library. However, for the results to be more independent of this environment, the efficiency of the algorithms will not be measured in time but in the number of distances to

be computed. We believe that this metric provides a better indicative of the theoretical efficiency of the algorithms.

4.1. Datasets and representations

In order to exhaustively assess the proposed method, we have considered a set of eight data collections that remarkably differ in their number of samples, classes, and features. A 5-fold cross-validation partitioning has been implemented, being the different classes equally represented in each of the partitions. At each experiment, one fold is used as test set, whereas the rest are used for training. For all cases, the Euclidean distance has been considered as distance measure for the kNN classifier.

The precise datasets considered are the *United States Postal Service* (USPS) digit dataset [43] that consists of images of 16x16 pixels of single handwritten digits, the *Handwritten Online Musical Symbol* (HOMUS) dataset [44] with images of 40x40 pixels of isolated handwritten music shapes, the *NIST SPECIAL DATABASE* (NIST) [45] of handwritten characters from which a subset of the upper case ones was randomly selected, the MNIST collection that contains images of 28x28 pixels representing isolated handwritten digits, and four datasets of the UCI repository [46]: the *Gisette* dataset of isolated handwritten digits that focuses on exclusively separating digits '4' and '9', the *Letter* collection that contains handwritten examples of the capital characters from the English language, the *Landsat* dataset that comprises satellite images analyzed in four spectral bands in image neighborhoods of 3x3 pixels, and the *Pendigits* compilation of handwritten isolated digits. Table 1 summarizes the information related to the datasets in terms of their number of samples, classes, descriptors, and optimal clustering configuration.

For obtaining the NC of each dataset we initially define a series of DNN architectures. The specific configuration of each network is experimentally determined—inspired by topologies proposed in similar tasks. The idea, however, is that the base DNNs perform similarly close to the state of the art in each dataset. The precise DNN architectures considered for each dataset are listed in

Table 1: Summary of the datasets used in the assessment of the proposed method in terms of their number of samples, classes, and features. Notation ($c \times w \times h$) denotes that the features are directly the pixels of the image, representing c the number of channels, w the width, and h the height in pixels. The *Optimal clustering* column represents the theoretical optimum number of clusters for each set.

Dataset	Samples	Optimal clustering	Classes	Features
USPS	9298	96	10	256 ($1 \times 16 \times 16$)
HOMUS	15200	123	32	1600 ($1 \times 40 \times 40$)
NIST	44951	212	26	1024 ($1 \times 32 \times 32$)
MNIST	70000	265	10	784 ($1 \times 28 \times 28$)
Gisette	7000	84	2	5000
Letter	2000	141	26	16
Landsat	6435	80	6	36 ($4 \times 3 \times 3$)
Pendigits	10992	105	10	16
Average	22985	138	-	-

Table 2. While these architectures differ in a number of parameters, all of them comprise a final fully-connected layer of 128 neurons. Therefore, the NC representation considered always consist of a 128-dimensional vector. We checked that this length was the one that depicted a generally better performance taking into account the data collections considered.

The learning of the network weights is performed during 200 epochs by means of stochastic gradient descent [47] with a mini-batch size of 128 samples, considering the adaptive learning rate proposed by Zeiler [48] (with default parameterization).

4.2. Evaluation methodology

For quantitatively assessing the classification goodness of the system we consider the F-measure (F_1) class-wise measure. Taking one class at a time as reference, this metric summarizes the correctly classified elements (*True Positive*, TP), the misclassified instances from the other classes as being from the reference one (*False Positive*, FP), and the misclassified elements from the ref-

Table 2: DNN architectures considered for mapping the initial data features of the datasets to their respective NC representations. Notation $Conv(f, w, h)$ stands for a layer with f convolution operators of size $w \times h$ pixels, $FC(n)$ represents a fully-connected layer of n neurons, $Drop(d)$ implements a dropout stage with a value of d %, $MaxPool(w, h)$ stands for the max-pooling operator of dimensions $w \times h$ pixels, and $UpSamp(w, h)$ represents an up-sampling process of size $w \times h$ pixels.

Dataset	Network architecture						
	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
USPS MNIST	Conv(32x3x3)	Conv(32x3x3) MaxPool(2x2) Drop(0.25)	FC(128) Drop(0.5)				
Letter Pendigits	FC(512) Drop(0.2)	FC(1024) Drop(0.2)	FC(512) Drop(0.2)	FC(256) Drop(0.2)	FC(128) Drop(0.2)		
Landsat	Conv(64x1x1) UpSamp(2x2) Drop(0.3)	Conv(64x2x2) UpSamp(2x2) Drop(0.3)	Conv(64x2x2) UpSamp(2x2) MaxPool(2x2) Drop(0.3)	FC(256) Drop(0.3)	FC(128) Drop(0.3)		
Gisette	FC(4096) Drop(0.5)	FC(2048) Drop(0.5)	FC(1024) Drop(0.5)	FC(512) Drop(0.5)	FC(256) Drop(0.5)	FC(128) Drop(0.5)	
HOMUS NIST	Conv(256x3x3) MaxPool(2x2) Drop(0.2)	Conv(128x3x3) MaxPool(2x2) Drop(0.2)	Conv(128x3x3) Drop(0.2)	Conv(64x3x3) Drop(0.2)	FC(512) Drop(0.1)	FC(256) Drop(0.1)	FC(128) Drop(0.1)

erence class as being examples of one of the others (*False Negative*, FN) in a single value as:

$$\mathrm{F}_1 = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \quad .$$

In addition, as the proposed method is based on improving the classification performance at the expense of including additional instances in the process, we also analyze the efficiency of the model. When it comes to measuring the efficiency of the kNN search, it is typical to use the number of distances needed throughout the process. This is done for several reasons. On the one hand, it is less dependent on the underlying computer in which the experiments are conducted. On the other hand, it unifies the criterion when using heterogeneous datasets, with a different number of features or even different distances (Euclidean, Mahalanobis, Hamming, etc.). In our case, in addition, NC repre-

sentations contain always 128 values, thereby being equivalent for all datasets, and less than the number of features of the original space (with exceptions that are negligible).

Hence we may be able to assess the improvement achieved in relation to the total set size measured as the number of distances computed.

Note that classification performance and set size are generally opposing goals as improving one of them generally implies a deterioration of the other one, being then difficult to select an optimal number of clusters c that optimizes the task. In this regard, additional insights may be gained by assessing this proposal from a *Multi-objective Optimization Problem* (MOP) perspective in which both the number of distances to compute (or training set size) and the classification performance are meant to be optimized at the same time. Usually this evaluation is carried out by means of the non-dominance concept: one solution (in this case, tuple of the classification performance and number of distances computed for a number of clusters c) is said to dominate another if, and only if, it is better or equal in each goal function and, at least, strictly better in one of them. The Pareto frontier stands for the set of all non-dominated elements and represents the different optimal solutions to the MOP. Each of these solutions, which is referred to as Pareto-optimal configuration, are considered the best solutions to the problem without any particular priority among them.

4.3. Comparative approaches

In order to comparatively assess the performance of the proposed ckNN+ strategy we shall study and compare the behavior obtained when considering different configurations. For that we shall initially analyze the improvement that the use of the NC representation implies by comparing it to the case when using the initial feature representations of the datasets.

Additionally, we consider different values of the number of clusters c to study the influence of that parameter. More precisely, the values studied are 1, 10, 15, 20, 25, 30, 100, 500, and 1000. For all these cases we also examine the implications of the cluster augmentation process, that is ckNN+ against ckNN,

both in terms of performance and number of distances. Note that the case of $c = 1$ is equivalent to performing an exhaustive kNN search as all instances are grouped into one single cluster.

Finally, we contemplate different values of the number of neighbors k for both the cluster augmentation and the final classification stage. More precisely, the set of values considered are 1, 3, 5, and 7.

5. Results

This section introduces the different experimental results for assessing the performance of the proposed strategy. For that, we perform three different experiments: a first one devoted to thoroughly assess the behavior of the method proposed in the paper; a second one in which we compare our proposal to other existing strategies for improving the deficiencies found in the kNN classifier; and a third set of experiments in which we assess the influence of the use of Neural Codes in the goodness of the clustering process.

5.1. Evaluation of the proposed method

This first part of the section analyzes the performance of the proposed method when considering the experimental scheme introduced in Section 4.2. These results are shown in Table 3 as the average performance values and percentage of distances for all datasets considered. Labels *Original* and *NC* denote the use of either the initial feature representation or the neural codes, respectively. With the same idea, *ckNN* and *ckNN+* represent the cases when classification is done using the partitions obtained with the initial clustering process or with the augmented one, respectively. Values in bold represent the non-dominated solutions obtained.

Let us initially focus on the case with a single cluster (i.e., clustering process configured to $c = 1$), which shall act as a reference to compare with throughout this analysis section. Note that for this case, no difference in terms of performance or number of distances computed may be appreciated between the

Table 3: Average results across the datasets considered for the F_1 and number of distances metrics for each configuration given by the number of clusters c and number of neighbors k. *Original* and *NC* stand for the initial feature space and the neural code representations, respectively. *ckNN* and *ckNN+* represent the result of the clustering process and the cluster augmentation stage, respectively. Non-dominated elements are highlighted.

c	k	F_1 (%)				Distances (%)			
		Original		NC		Original		NC	
		ckNN	ckNN+	ckNN	ckNN+	ckNN	ckNN+	ckNN	ckNN+
1	1	**90.0**	90.0	98.9	98.9	**100.0**	100.0	100.0	100.0
	3	89.5	89.5	**99.1**	99.1	100.0	100.0	**100.0**	100.0
	5	89.4	89.4	99.1	99.1	100.0	100.0	100.0	100.0
	7	89.1	89.1	99.1	99.1	100.0	100.0	100.0	100.0
10	1	**88.8**	**88.9**	98.7	98.7	**12.6**	**14.0**	12.3	12.6
	3	87.7	88.4	98.8	**98.9**	12.6	16.8	12.3	**13.1**
	5	87.2	88.2	98.7	98.9	12.6	19.7	12.3	13.9
	7	86.6	88.0	98.7	98.9	12.6	22.4	12.3	14.9
15	1	88.3	**88.6**	98.7	98.7	8.2	**9.3**	8.5	8.7
	3	87.0	87.9	98.8	**98.9**	8.2	11.7	8.5	**9.2**
	5	86.3	87.8	98.7	98.9	8.2	14.0	8.5	9.8
	7	85.6	87.4	98.6	98.9	8.2	16.2	8.5	10.6
20	1	**88.2**	**88.4**	98.7	98.7	**6.1**	**7.0**	6.6	6.8
	3	86.8	87.8	98.7	**98.9**	6.1	9.0	6.6	**7.3**
	5	86.0	87.5	98.6	98.9	6.1	10.9	6.6	7.9
	7	85.1	87.2	98.6	98.9	6.1	12.8	6.6	8.6
25	1	88.0	88.2	98.7	98.7	5.1	5.9	5.0	5.2
	3	86.6	87.6	98.7	**98.8**	5.1	7.5	5.0	**5.7**
	5	85.7	87.3	98.6	98.8	5.1	9.2	5.0	6.2
	7	84.9	86.9	98.6	98.8	5.1	10.8	5.0	6.8
30	1	**87.8**	**88.1**	98.7	98.7	**4.4**	**5.1**	4.2	4.4
	3	86.3	87.4	**98.7**	98.8	4.4	6.6	**4.2**	4.9
	5	85.4	87.1	98.6	98.8	4.4	8.1	4.2	5.4
	7	84.6	86.8	98.6	98.8	4.4	9.6	4.2	6.0
100	1	**87.1**	**87.2**	**98.7**	98.7	**2.3**	**2.6**	**2.3**	2.4
	3	85.3	86.4	98.7	**98.8**	2.3	3.3	2.3	**2.7**
	5	84.3	86.1	98.7	98.8	2.3	3.9	2.3	3.1
	7	83.5	85.6	98.7	98.8	2.3	4.5	2.3	3.4
500	1	86.7	86.8	98.7	98.7	5.4	5.5	5.4	5.4
	3	84.8	85.7	98.8	98.8	5.4	5.7	5.4	5.6
	5	84.0	85.4	98.7	98.8	5.4	5.9	5.4	5.7
	7	83.3	84.9	98.7	98.8	5.4	6.1	5.4	5.8
1000	1	86.9	87.0	98.7	98.7	10.3	10.4	10.3	10.3
	3	85.2	85.9	98.8	98.9	10.3	10.5	10.3	10.4
	5	84.5	85.4	98.8	98.8	10.3	10.7	10.3	10.5
	7	83.9	85.0	98.7	98.9	10.3	10.8	10.3	10.6

initial and augmented clustering schemes (labeled as ckNN and ckNN+ respectively) as the entire training set is considered for the classification (distances computed during the classification stage constitute the maximum expected). As it may be checked, when considering the *Original* feature space, performance results are considerably elevated with values around $F_1 = 90$ % for the different configurations of the kNN algorithm. In addition, the fact that the maximum performance is achieved for the $k = 1$ configuration and that increasing k lowers the performance somehow suggests that datasets may be hardly noisy, but there is still room for improvement. In this context, the use of the *Neural Code* (NC) representation entails a clear advantage in terms of performance: for all cases, performance results are improved in up to a 10 %, which leads to error values of just 1 % in the F_1 metric.

While the aforementioned configuration stands for the one in which the best accuracy results may be achieved, this is also the most computationally complex situation since all instances are examined at the classification stage. Thus, it would be expectable that the use of the clustering-based search (that is, ckNN) should provide a more efficient method (less instances are queried during the classification stage) at the expense of a decrease in the classification performance. We shall now examine the obtained results to verify this premise.

As it may be checked, the ckNN cluster-based strategy entails a decrease in the performance of the system. For instance, focusing on the $k = 1$ for the *Original* feature representation: in this situation the performance degrades from a value of $F_1 = 90$ % for $c = 1$ to a value of $F_1 = 86.9$ % with $c = 1000$ clusters; however, paired with that performance decrease, the number of distances is also reduced from the exhaustive search in the former case to values of roughly 10 % of the distances for the latter. Note that global minimum in the number of distances (around 2 % of the total computation) is achieved when selecting $c = 100$, which is the closest value to the $c = 138$ that results from averaging the optimal number of clusters for each of the datasets studied (cf. Table 1).

In order to tackle this decrease in the performance we consider the ckNN+ cluster augmentation process proposed. As it can be observed, the inclusion

of the neighboring instances for each of the clusters improves the classification performance of the system: for instance, performance in the case when considering $c = 20$ clusters and $k = 7$ neighbors increases from, roughly, a value of $F_1 = 85$ % to a value of $F_1 = 87$ % at the expense of computing approximately 13 % of the total number of distances instead of a 6 %. In general terms, this cluster augmentation process entails improvements up to a 2 % in the performance with increases of as much as 10 % in the number of distances.

Considering now the NC feature representation, the situation subtly changes. The first point to comment is that, as it happened in the case of $c = 1$, the use of NC entails a remarkable improvement with respect to the use of the raw features of each dataset, reaching an improvement around 10 % and 15 % in the F_1, depending on the particular case. However, the most remarkable point to highlight is that, with these NC representations, the performance of the classifier stays relatively steady independently of the number of clusters considered. This fact suggests that the NC representation involves features that properly gather instances representing the same class, thus the clustering process does not appreciably affect the overall performance but still the number of distances is remarkably reduced. For example, with 1000 clusters and comparing with the exact search (c = 1), there is an accuracy loss of only 1% with NC, instead of the 5% that occurs with the original representation.

The cluster augmentation process also entails an additional improvement on the performance of the system that, while less accused than in the case when considering the raw feature space, also supposes an increase of, as much, a 2 % in the number of distances.

Having analyzed the classification performance and computational reduction as stand-alone measures of merit, we shall now study them jointly considering the MOP scenario. For the sake of clarity, Fig. 3 graphically shows the results obtained: Pareto frontiers are obtained separately for each feature representation (raw features as *Original* and neural codes as *NC*); initial and augmented clusters (ckNN and ckNN+, respectively) are represented by the shape and color of the point as depicted in the legend; non-dominated elements are highlighted

and their configuration in terms of the number of clusters c and amount of neighbors k is shown.

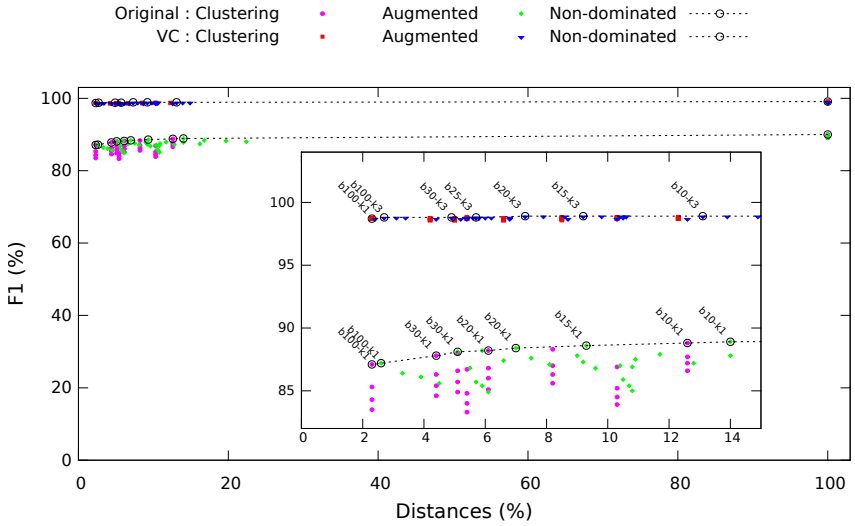


Figure 3: Graphical representation of the results obtained. Non-dominated elements are highlighted.

As a first point to comment is that this graphical representation clearly shows the advantage of the use of the NC if compared to the case of the raw feature: while the latter representation shows a tendency that approaches to a value of $F_1 = 90$ %, the former method shows a similar asymptotic trend but approaching a value $F_1 = 99$ %. Also, as previously commented, this representation allows to easily check that the use of the ckNN+ cluster augmentation process entails a larger performance improvement in the case when considering raw features rather than in the NC one.

The analysis of the non-dominated elements depicts some additional conclusions to the aforementioned ones. Focusing on the NC representation case as it is the one achieving the best overall performance, it can be checked that most solutions in the Pareto frontier are configurations of the augmented cluster approach. Such results point out that this enlargement process applied to the initial clusters obtained endows the system with the precise additional in-

stances necessary for the proper compromise between classification performance and computational complexity, measured as the number of distances computed at the classification stage.

Finally, to rigorously analyze the results obtained and derive strong conclusions out of them, we now perform a statistical significance analysis by considering the non-parametric Wilcoxon signed-rank test [49]. More precisely, the idea is to assess whether the improvement observed in the classification performance with the use of the ckNN+ cluster augmentation and the NC representations is statistically relevant. Thus, this analysis shall not consider the computational complexity of the strategy at issue.

Table 4 shows the statistical comparison of the classification performance of the ckNN+ cluster-augmentation strategy against the initial ckNN cluster-based method for the raw initial features (*Original*) and the NC representation (*NC*). For each cluster configuration c, the value of k that maximizes the classification rate has been selected. Configuration $c = 1$ is obviated since results for both strategies are equivalent. Statistical significance has been fixed to a value $p < 0.01$.

Table 4: Results of the Wilcoxon test comparing the classification performance of the cluster-augmentation strategy against the initial cluster-based method for the raw initial features (*Original*) and the Neural Codes representation (*NC*). Symbols ✓and its absence represent that the cluster-augmentation strategy (ckNN+) significantly improves or not over the initial cluster-based method (ckNN), respectively. The case $c = 1$ is obviated since results for both strategies are equivalent. A significance level $p < 0.01$ has been considered for the analysis.

Feature space	Number of clusters c							
	10	15	20	25	30	100	500	1000
Original	✓	✓	✓	✓	✓	✓		
NC	✓	✓	✓	✓	✓	✓	✓	✓

The results of the Wilcoxon test show that, in general, the ckNN+ cluster augmentation process significantly improves the initial ckNN cluster-based method, for both the raw feature and NC representation spaces. The only cases in which this assertion is not accomplished are the configurations of $c = 500$

and $c = 1000$ clusters when considering the raw initial features in which the improvement is not as sharp as in the rest of the cases. This fact clearly states the advantage and usefulness of the proposed combined strategy of NC representations and ckNN+ cluster augmentation compared to the regular ckNN cluster-based kNN search.

Finally, note that this analysis has considered a statistical significance threshold of $p < 0.01$, which is more restrictive than the typical threshold of $p < 0.05$ commonly found in the Pattern Recognition field. This decision has been motivated by the fact that with the threshold $p < 0.05$ all cluster-augmentation cases (ckNN+) significantly improved over the initial clustering process (ckNN) for both the *Original* and *NC* cases. Thus, using the more restrictive threshold of $p < 0.01$ provided additional insights that are not observed in the former case.

5.2. Comparison with other strategies for the kNN limitations

Once we have studied the performance of the proposed method we shall comparatively assess it against other existing strategies which also aim at improving the aforementioned limitations of the kNN classifier. For that we have selected a set of representative strategies from the different optimization families introduced in Section 2.1:

- **FSS**: In terms of this family of space partitioning techniques we have selected the k-d tree [19] and ball tree [21] methods. We have configured them to examine $10, 20, 30,$ and 40 prototypes in each leaf node.

- **ASS**: As of approximate search approaches, we have considered the use of hashing techniques. Since these techniques also learn a mapping out of data, its approach is similar to ours, and so they are suitable for comparison. Specifically, we choose LSH forest algorithm [31], Spectral Hashing [32] and Product Quantization [33]. For the former, we tweaked the algorithm to consider $10, 20, 30,$ and 40 trees in the search process.

- **DR**: For this particular family of approaches we have assessed two different options: on the one hand, we consider the Reduction through Homogeneous Clusters (RHC) algorithm [1], which is interesting for this paper because it is based on clustering; on the other hand, we test the meta-algorithm kNNc [30], which receives a DR method as parameter for its operation. For this latter option, the following DR strategies have been considered:
  - Condensing Nearest Neighbor [24], Multi-Edit Condensing Nearest Neighbor [50], and Fast Condensing Nearest Neighbor [51].
  - Editing Nearest Neighbor [52] and Multi-Editing [53].
  - Farther Neighbor and Nearest to Enemy rank methods [54].
  - Decremental Reduction Optimization Procedure 3 [55].
  - Iterative Case Filtering Algorithm [56].
  - Cross-generational elitist selection, Heterogeneous recombination, and cataclysmic mutation algorithm introduced in [57] as a representative of Genetic Algorithms for DR.

For the comparative, all the methods consider the initial feature space of the data collections. As in the previous section, we have tested the influence of the parameter k with values $k = 1, 3, 5,$ and 7.

The results obtained by these algorithms considering the same evaluation methodology as in the previous section (data collections, cross-validation schemes, and metrics) together with the results obtained by the ckNN+ method are shown in Fig. 4. Non-dominance analysis is performed separately for both our approach and the rest of the algorithms.

As it can be checked, the results obtained by the proposed ckNN+ method outperform the rest of the strategies considered in both classification performance and number of distances. In terms of classification performance, the family which achieved the closest results to the ones obtained by the ckNN+ is the FSS one. Nevertheless note that, while this FSS family of methods achieved

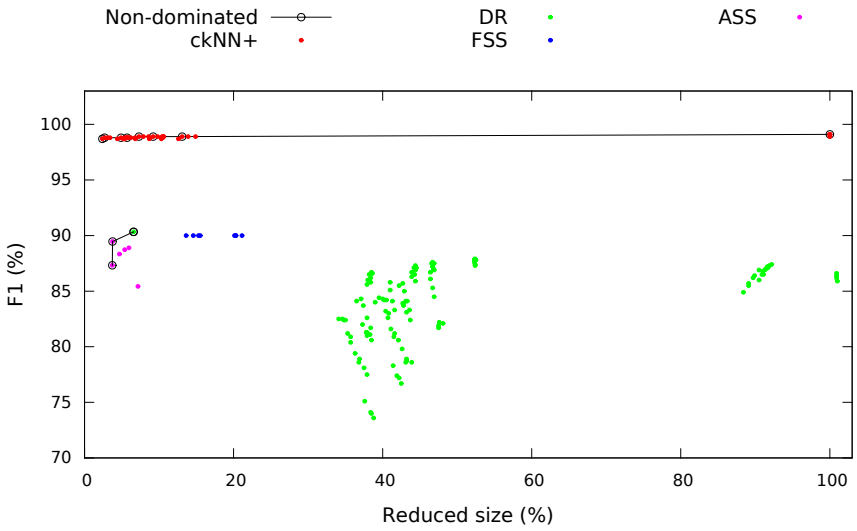


Figure 4: Comparison of the proposed ckNN+ strategy against other kNN strategies in terms of number of distances computed and classification performance. Non-dominated elements are highlighted for each kNN strategy.

results of $F_1 \approx 90$ %, these results are noticeably far from the performance of the ckNN+ (almost a 10 % in the F_1 measure).

As regards the number of distances computed, the ASS family showed competitive results compared to the ckNN+ method. Distances computed are around a 5 % of the total number of comparisons to perform. Nevertheless, this approximated search is also reflected on the fact that the accuracy of the classifier is decreased with respect to the FSS methods.

Finally, most of the DR strategies are clearly not competitive against any of the other strategies considered since the number of distances computed is larger than the ones computed by the other strategies paired with lower classification rates. However, one of the them, namely RHC, is able to achieve very good results in both accuracy and efficiency, and so it belongs to the non-dominance front of the compared methods.

5.3. Impact of Neural Codes on clustering

As a final analysis of the proposed strategy, in this section we focus on studying the clustering performed as a first step of the ckNN+. More precisely, our intention is to measure the goodness produced by the use of NC within this stage. To carry out this analysis, we consider the following set of measures, commonly used for this purpose:

Silhouette Coefficient [58] (SCoeff) takes into account both the average intra-cluster distance a and the average extra-cluster distance b. Then, a coefficient is computed as

$$\frac{b-a}{\max(a,b)}.$$

Values close to 1 represent compact clusters, whereas negatives values are obtained when samples are assigned to wrong clusters.

Calinski-Harabaz Index [59] (CHI) considers the between-clusters dispersion and the within-cluster dispersion, depicting a score that is higher when clusters are dense and well separated.

Homogeneity Score [60] (HSc) is a supervised metric that indicates the average homogeneity (ratio of samples from the same class) of the clusters.

Completeness Score [60] (CSc) is also a supervised metric that approaches to 1 as the completeness (ratio of the samples of the same class that are assigned to the same clusters) is increased.

Table 5 shows the evaluation performed, comparing the process of clustering with the original characteristics or NC. On the one hand, it is observed that for each comparison, and for every metric, the clustering performed with NC is better than the corresponding one with original features. On the other hand, it is also observed that, according to the metrics SCoeff, HSc, and CSc, the process is degraded as the number of clusters increases. This is reflected even in the case of considering NC. However, the CHI measure reflects, unlike the

previous ones, that increasing the number of clusters is beneficial as long as NC is used (the opposite occurs with the original features), reinforcing once again the use of this type of representation within the proposed approach.

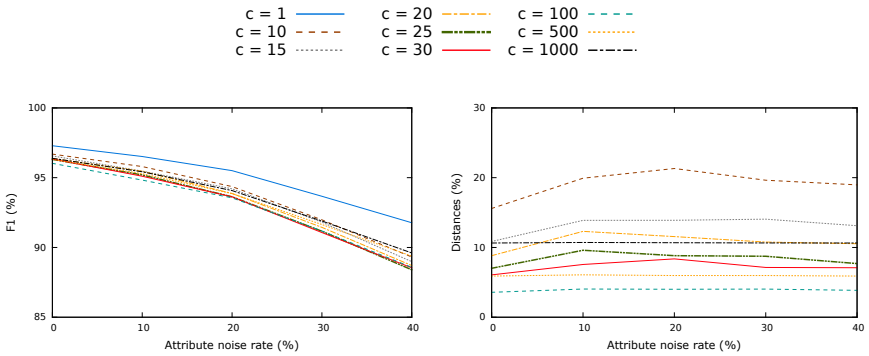
Table 5: Evaluation of the quality of the clustering performed in the first step of ckNN+ as regards the use of the original features (Original) or the Neural Codes (NC) representation. Values represent the average with respect to the considered datasets. The higher the values, the better the performance of the clustering.

c	SCoeff		CHI		HSc		CSc	
	Original	NC	Original	NC	Original	NC	Original	NC
10	0.16	0.38	1395.66	6961.50	0.46	0.78	0.47	0.78
15	0.14	0.38	1147.36	8531.80	0.45	0.73	0.46	0.73
20	0.13	0.37	980.45	9805.35	0.43	0.70	0.44	0.70
25	0.13	0.35	872.00	11288.86	0.42	0.67	0.43	0.67
30	0.12	0.35	793.94	12676.63	0.42	0.65	0.43	0.65
100	0.10	0.23	406.63	23611.29	0.37	0.50	0.39	0.50
500	0.09	0.16	159.17	15659.51	0.32	0.37	0.34	0.38
1000	0.09	0.14	108.00	11731.73	0.30	0.34	0.33	0.35

Results reported above, however, only reflect average values. In order to minimize the possibility that the differences are due to chance variation, we perform again the Wilcoxon signed-rank test. We considered the 8 independent results (one per number of clusters c) to perform these tests. It resulted in p-values below 0.01 in all pairwise comparisons assuming that NC provides better metrics than original features. Therefore, the use of NC leads to better clustering processes—according to the metrics considered—with an alpha confidence level of 99%.

5.4. Evaluation in the presence of attribute noise

Given that the clustering process is a key aspect of our proposal, this section studies the effect that noise in the instances has on the overall success of the task. In this case, the interest is in the noise at the attribute level, since it is the one that influences the composition of the clusters.



(a) Accuracy (b) Efficiency

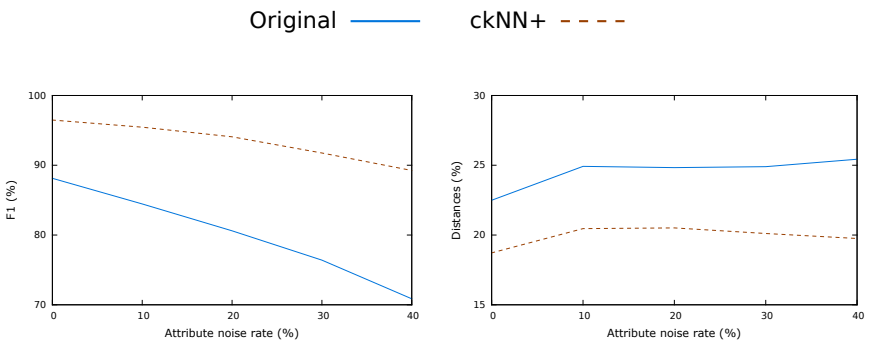
Figure 5: Performance of the ckNN+ with respect to the number of clusters in the presence of attribute noise.

We perform this evaluation as in the work of Zu and Whu [61], in which attribute noise is generated synthetically to study its effects in a controlled environment. The process consists in randomly disturbing the attributes of the data set. This perturbation modifies an attribute randomly within the range of the possible values of that attribute in the domain of the problem. The process is guided by the parameter *attribute noise rate*, which indicates the likelihood that an attribute is modified.

Figure 5 shows the accuracy and efficiency trends of the ckNN+ in the presence of attribute noise, with rates of 0, 10, 20, 30, and 40 (%). In each case, the best value obtained for the different values of k is presented.

As expected, Fig. 5a reports that the accuracy degrades as the noise increases. However, it can be seen that the tendencies are similar for the original kNN ($c = 1$). Therefore, we can conclude that performance is equally affected than in the brute-force approach.

Concerning efficiency, Fig. 5b shows the average number of distances (in percentage with respect to $c = 1$, which computes 100 % of the distances) needed to classify a sample. Here a clear trend is observed with respect to parameter c. In the case of few clusters ($c = 10, 15, 20, 25, 30$), the efficiency seems to slightly degrade as the noise at the attribute level increases. However,



(a) Accuracy (b) Efficiency

Figure 6: Comparison of performance between the use of original features and ckNN+ in the presence of attribute noise.

when the number of clusters is high ($c = 100, 500$), the scheme is much more robust to this phenomenon.

On the other hand, we depict in Fig. 6 a comparison of the degradation suffered by our method and that of the original representation of the data. For the sake of clarity, both results are shown with the average values of the different c.

It is observed that our approach is much more robust with respect to accuracy (Fig. 6a), since the trends are much more stable. This is because the data mapping onto NC alleviates the effects caused by noise at the attribute level. Obviously, if the noise is mitigated, the classification is more reliable; in addition, as presented in the previous Section 5.3, the use of more appropriate features (in this case, reducing the effects of noise) contributes to a better distribution of the data in the clusters, and therefore a higher average efficiency in the search is attained (Fig. 6b).

6. Conclusions and Future Work

The k-Nearest Neighbor (kNN) rule represents a widely considered supervised classification scheme due to its conceptual simplicity and straight-forward implementation as well as its statistical properties regarding its error bounds and

classification performance. As a representative example of instance-based classification, kNN does not derive a model out the initial training data considered. This fact constitutes a disadvantage when large-scale datasets are considered as all instances of the training data have to be queried whenever a new prototype has to be classified.

In this paper we take as initial point a two-stage search strategy based on clustering for lowering the computational cost of the kNN classifier. The idea is that the clustering process distributes the prototypes in the training data in a set of groups and their centroids are retrieved; a query to be classified is assigned to the cluster according to its closest centroid and the eventual class is retrieved applying the kNN rule using the elements in the cluster.

Given that the aforementioned strategy usually entails a decrease in the classification performance, two modifications are proposed to that scheme: on the one hand, we propose a strategy for improving the classification rate by solving issues with instances located close to the cluster boundaries by enlarging their size; on the other hand, we consider the use of Deep Neural Networks for the automatic extraction of feature-based representations (Neural Codes), which properly gather instances of the same class so that they fall within the same cluster. Results using a collection of several datasets show that the combined use of these two strategies entails a considerable reduction in the number of distances performed at the classification stage with a significant improvement in the classification performance when compared to the basic kNN rule. Additionally, the proposed scheme has been exhaustively compared to a set of well-known strategies for solving the aforementioned kNN deficiencies. Results obtained reinforce the conclusions previously gathered about the competitive performance and low computational as, for both performance and computational cost criteria, the proposed methods outperforms the rest of the strategies considered. In addition, the use of Neural Codes has been evaluated according to the quality of the clusters obtained and tolerance to noise at attribute level. In both cases, it has been empirically demonstrated that the performance improvement is remarkable.

Future works consider further analysis of Deep Neural Networks for deriving the optimal feature representations by studying other loss functions and more advanced architectures. Also, given the great separability achieved by the deep feature representations, it seems interesting to apply Prototype Selection and Generation techniques for the kNN classifier as they may remarkably reduce the number of instances in the training set with minimal accuracy losses.

Acknowledgments

This work has been supported by the Spanish Ministerio de Economía y Competitividad through Project TIMuL (No. TIN2013-48152-C2-1-R supported by EU FEDER funds), the Spanish Ministerio de Educación, Cultura y Deporte through a FPU Fellowship (Ref. AP2012–0939), and by the Universidad de Alicante through the FPU program (UAFPU2014–5883) and through the Instituto Universitario de Investigación Informática (IUII).

References

[1] S. Ougiaroglou, G. Evangelidis, RHC: a non-parametric cluster-based data reduction for efficient k-NN classification, Pattern Analysis and Applications 19 (1) (2016) 93–109.

[2] L. Yang, Q. Zhu, J. Huang, D. Cheng, Adaptive edited natural neighbor algorithm, Neurocomputing 230 (2017) 427–433.

[3] S. Zhang, X. Li, M. Zong, X. Zhu, R. Wang, Efficient kNN Classification With Different Numbers of Nearest Neighbors, IEEE Transactions on Neural Networks and Learning Systems [Online] (2017) 1–12.

[4] J. Maillo, S. Ramírez, I. Triguero, F. Herrera, kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data, Knowledge-Based Systems 117 (2017) 3–15.

[5] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, J. M. Benítez, F. Herrera, Nearest Neighbor Classification for High-Speed Big Data Streams Using Spark, IEEE Transactions on Systems, Man, and Cybernetics: Systems.

[6] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE transactions on information theory 13 (1) (1967) 21–27.

[7] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[8] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, IEEE Transactions on Neural Networks 13 (2) (2002) 415–425.

[9] T. M. Mitchell, Machine Learning, McGraw-Hill, Inc., 1997.

[10] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, 2nd Edition, John Wiley & Sons, New York, NY, 2001.

[11] X. Wu, X. Zhu, G.-Q. Wu, W. Ding, Data mining with big data, IEEE Trans. on Knowl. and Data Eng. 26 (1) (2014) 97–107.

[12] Z. Deng, X. Zhu, D. Cheng, M. Zong, S. Zhang, Efficient knn classification algorithm for big data, Neurocomputing 195 (2016) 143–148.

[13] F. Huang, Y. LeCun, Large-scale learning with SVM and convolutional nets for generic object categorization, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2006, Vol. 1, 2006, pp. 284–291.

[14] A. S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN Features Off-the-Shelf: An Astounding Baseline for Recognition, in: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW '14, IEEE Computer Society, Washington, DC, USA, 2014, pp. 512–519.

[15] Y. Tang, Deep learning using linear support vector machines, arXiv preprint arXiv:1306.0239.

[16] P. Jain, B. Kulis, I. S. Dhillon, K. Grauman, Online metric learning and fast similarity search, in: Proceedings of the 21st International Conference on Neural Information Processing Systems, NIPS'08, Curran Associates Inc., USA, 2008, pp. 761–768.

[17] S. García, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, Springer, 2015.

[18] J. Wang, H. T. Shen, J. Song, J. Ji, Hashing for similarity search: A survey, arXiv preprint arXiv:1408.2927.

[19] J. H. Friedman, J. L. Bentley, R. A. Finkel, An Algorithm for Finding Best Matches in Logarithmic Expected Time, ACM Transactions on Mathematical Software 3 (3) (1977) 209–226.

[20] E. Vidal, An algorithm for finding nearest neighbours in (approximately) constant average time, Pattern Recognition Letters 4 (3) (1986) 145–157.

[21] T. Liu, A. W. Moore, A. Gray, Efficient exact k-nn and nonparametric classification in high dimensions, in: Proceedings of the 16th International Conference on Neural Information Processing Systems, MIT Press, 2003, pp. 265–272.

[22] P. Ciaccia, M. Patella, P. Zezula, M-tree: An efficient access method for similarity search in metric spaces, 1997, pp. 426–435.

[23] L. Nanni, A. Lumini, Prototype reduction techniques: A comparison among different approaches, Expert Systems with Applications 38 (9) (2011) 11820–11828.

[24] P. Hart, The condensed nearest neighbor rule (corresp.), IEEE Transactions on Information Theory 14 (3) (1968) 515–516.

[25] S. Garcia, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: Taxonomy and empirical study, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3) (2012) 417–435.

[26] J. Derrac, C. Cornelis, S. García, F. Herrera, Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection, Information Sciences 186 (1) (2012) 73–92.

[27] J. Hamidzadeh, R. Monsefi, H. S. Yazdi, Irahc: instance reduction algorithm using hyperrectangle clustering, Pattern Recognition 48 (5) (2015) 1878–1889.

[28] N. García-Pedrajas, A. De Haro-García, Boosting instance selection algorithms, Knowledge-Based Systems 67 (2014) 342–360.

[29] C.-F. Tsai, W. Eberle, C.-Y. Chu, Genetic algorithms in feature and instance selection, Knowledge-Based Systems 39 (2013) 240–247.

[30] J. Calvo-Zaragoza, J. J. Valero-Mas, J. R. Rico-Juan, Improving kNN multi-label classification in prototype selection scenarios using class proposals, Pattern Recognition 48 (5) (2015) 1608–1622.

[31] M. Bawa, T. Condie, P. Ganesan, Lsh forest: self-tuning indexes for similarity search, in: Proceedings of the 14th international conference on World Wide Web, ACM, 2005, pp. 651–660.

[32] Y. Weiss, A. Torralba, R. Fergus, Spectral Hashing, in: Advances in Neural Information Processing Systems, 2008, pp. 1753–1760.

[33] H. Jegou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, IEEE transactions on pattern analysis and machine intelligence 33 (1) (2011) 117–128.

[34] M. Muja, D. G. Lowe, Scalable nearest neighbor algorithms for high dimensional data, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (11) (2014) 2227–2240.

[35] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[36] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, S. Carlsson, Factors of transferability for a generic convnet representation, IEEE transactions on pattern analysis and machine intelligence 38 (9) (2016) 1790–1802.

[37] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems (NIPS), 2014, pp. 3320–3328.

[38] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, Neural Codes for Image Retrieval, Springer International Publishing, Cham, 2014, pp. 584–599.

[39] W. Ren, Y. Yu, J. Zhang, K. Huang, Learning convolutional nonlinear features for k nearest neighbor image classification, in: Pattern Recognition (ICPR), 2014 22nd International Conference on, IEEE, 2014, pp. 4358–4363.

[40] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Third Edition, Academic Press, Inc., Orlando, FL, USA, 2006.

[41] L. Rokach, A survey of clustering algorithms, in: Data Mining and Knowledge Discovery Handbook, 2nd ed., 2010, pp. 269–298. doi:10.1007/978-0-387-09823-4_14.

[42] D. Arthur, S. Vassilvitskii, K-means++: The advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007, pp. 1027–1035.

[43] J. Hull, A database for handwritten text recognition research, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (5) (1994) 550–554.

[44] J. Calvo-Zaragoza, J. Oncina, Recognition of Pen-Based Music Notation: the HOMUS dataset, in: Proceedings of the 22nd International Conference on Pattern Recognition (ICPR), Stockholm, Sweden, 2014, pp. 3038–3043.

[45] R.-A. Wilkinson, J. Geist, S. Janet, P.-J. Grother, et al., The first census optical character recognition system conference, Tech. rep., US Department of Commerce (1992). doi:10.18434/T4H01C.

[46] M. Lichman, UCI Machine Learning Repository (2013).
URL http://archive.ics.uci.edu/ml

[47] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[48] M. D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701.

[49] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[50] B. V. Dasarathy, J. S. Sánchez, S. Townsend, Nearest Neighbour Editing and Condensing Tools-Synergy Exploitation, Pattern Anal. Appl. (2000) 19–30.

[51] F. Angiulli, Fast Nearest Neighbor Condensation for Large Data Sets Classification, Knowledge and Data Engineering, IEEE Transactions on 19 (11) (2007) 1450–1464.

[52] D. L. Wilson, Asymptotic Properties of Nearest Neighbor Rules Using Edited Data, Systems, Man and Cybernetics, IEEE Transactions on SMC-2 (3) (1972) 408–421. doi:10.1109/TSMC.1972.4309137.

[53] P. A. Devijver, J. Kittler, Pattern recognition: A statistical approach, Prentice Hall, 1982.

[54] J. R. Rico-Juan, J. M. Iñesta, New rank methods for reducing the size of the training set using the nearest neighbor rule, Pattern Recognition Letters 33 (5) (2012) 654–660.

[55] D. R. Wilson, T. R. Martinez, Instance pruning techniques, in: Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 403–411.

[56] H. Brighton, C. Mellish, On the Consistency of Information Filters for Lazy Learning Algorithms, in: J. Żytkow, J. Rauch (Eds.), Principles of Data Mining and Knowledge Discovery, Vol. 1704 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1999, pp. 283–288. doi:10.1007/978-3-540-48247-5_31.

[57] J. R. Cano, F. Herrera, M. Lozano, On the Combination of Evolutionary Algorithms and Stratified Strategies for Training Set Selection in Data Mining, Appl. Soft Comput. 6 (3) (2006) 323–332. doi:10.1016/j.asoc.2005.02.006.

[58] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65.

[59] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Communications in Statistics-theory and Methods 3 (1) (1974) 1–27.

[60] A. Rosenberg, J. Hirschberg, V-measure: A conditional entropy-based external cluster evaluation measure, in: EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic, 2007, pp. 410–420.

[61] X. Zhu, X. Wu, Class noise vs. attribute noise: A quantitative study, Artificial Intelligence Review 22 (3) (2004) 177–210.