Class Representatives Selection in Non-metric Spaces for Nearest Prototype Classification

Jaroslav Hlaváč[1,2(✉)], Martin Kopp[1], Jan Kohout[1,3], and Tomá Skopal[2]

[1] TD&R Data Science, Cisco Systems, Prague, Czech Republic
hlavac.jaroslav@gmail.com
[2] Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
[3] TruU, Prague, Czech Republic

Abstract. The nearest prototype classification is a less computationally intensive replacement for the k-NN method, especially when large datasets are considered. Centroids are often used as prototypes to represent whole classes in metric spaces. Selection of class prototypes in non-metric spaces is more challenging as the idea of computing centroids is not directly applicable. Instead, a set of representative objects can be used as the class prototype.

In this paper, we present CRS, a novel memory and computationally efficient method that finds a small yet representative set of objects from each class to be used as prototype. CRS leverages the similarity graph representation of each class created by the NN-Descent algorithm to pick a low number of representatives that ensure sufficient class coverage. Thanks to the graph-based approach, CRS can be applied to any space where at least a pairwise similarity can be defined. In the experimental evaluation, we demonstrate that our method outperforms the state-of-the-art techniques on multiple datasets from different domains.

Keywords: Class Representation · Nearest Prototype Classification · Prototype Selection

1 Introduction

The k-NN classifiers are often used in many application domains due to their simplicity and ability to trace the classification decision to a specific set of samples. However, their adoption is limited by high computational complexity and memory requirements. Because contemporary datasets are often huge, containing hundreds of thousands or even millions of samples, computing similarity between the classified sample and the entire dataset may be computationally intractable.

In order to decrease computational and memory requirements, the nearest prototype classification (NPC) method is commonly used, c.f. [1–3]. In NPC, each class is represented by a *prototype*, that represents typical characteristics of

O. Pedreira and V. Estivill-Castro (Eds.): SISAP 2023, LNCS 14289, pp. 111–124, 2023.
https://doi.org/10.1007/978-3-031-46994-7_10

the class. The classified sample is then compared just to the prototypes instead of calculating similarity to the entire dataset. Therefore, the goal of prototype selection is to find a memory-efficient representation of classes such that classification accuracy is preserved while the number of comparisons is significantly reduced.

An intuitive prototype in a metric space can be a centriod. But even in metric spaces a centroid is often not an optimal solution because a single point does not represent the whole class well. Sometimes the centroid does not make sense and in non-metric spaces (also called distance spaces [4]) it is not defined. Such is the case in many application domains, where objects exist in space where only a pairwise (dis)similarity is defined, e.g., bioinformatics [5], biometric identification [6], or pattern recognition [7].

Our focus on non-metric spaces comes from the problem of behavioural clustering of network hosts [8], where we need to quickly assign a network host to a group of other hosts. A newly appearing network host in a computer network needs to be quickly assigned to a correct host group (or a new group must be created). The space we operate in is defined by the domains and IP addresses that the whole network has communicated with in the previous sliding time window (e.g. day). The similarity we use is expensive to compute (see [8] for details) as the dimension of the space is high and changes quickly.

Nevertheless, the problem of selecting a minimal number of representative samples is of more general interest. Only a few methods have been developed for non-metric scenarios, and to the best of our knowledge the only general (not domain-specific) approach is selection of small subset of objects to represent the whole class. The method is referred to as representative selection and the *representatives* (selected objects), are used as a prototype. Several recent methods capable of solving representatives selection on non-metric spaces exist (i.e. DS3 [9], δ-medoids [10]).

In this paper, we present a novel method to solve the problem of representative selection – Class Representatives Selection (CRS). CRS is a general method capable of selecting small yet representative subset of objects from a class to serve as its prototype. Its core idea is fast construction of an approximate reverse k-NN graph and then solving minimal vertex cover problem on that graph. Only a pairwise similarity is required to build the reverse k-NN graph, therefore application of CRS is not limited to metric spaces.

To show that CRS is general and domain-independent, we present an experimental evaluation on datasets from image recognition, document classification and network host classification, with appealing results when compared to the current state of the art. The code for CRS can be found at https://github.com/jaroslavh/ceres.

The paper is organized as follows. The related work is briefly reviewed in the next section. Section 3 formalises the representative selection as an optimization problem. The proposed method is described in detail in Sect. 4. The experimental evaluation is summarized in Sect. 5 followed by the conclusion.

2 Related Work

During the past years, significant effort has been made to represent classes in the most condensed way. The approaches could be categorized into two main groups.

The first group gathers all prototype generation methods [11], which create artificial samples to represent original classes, e.g. [12,13]. The second group contains the prototype selection methods. As the name suggests, a subset of samples from the given class is selected to represent it. Prototype selection is a well-explored field with many approaches, see, e.g. [14].

However, most of the current algorithms exploit the properties of the metric space, e.g., structured sparsity [15], l_1-norm induced selection [16] or identification of borderline objects [17].

When we leave the luxury of the metric space and focus on situations where only a pairwise similarity exists or where averaging of existing samples may create an object without meaning, there is not much previous work.

The δ-medoids [10] algorithm uses the idea of k-medoids to semi-greedily cover the space with δ-neighbourhoods, in which it then looks for an optimal medoid to represent a given neighbourhood. The main issue of this method is the selection of δ: this hyperparameter has to be fine-tuned based on the domain.

The DS3 [9] algorithm calculates the full similarity matrix and then selects representatives by a row-sparsity regularized trace minimization program which tries to minimize the rows needed to encode the whole matrix. The overall computational complexity is the most significant disadvantage of this algorithm, despite some proposed approximate estimation of the similarity matrix using only a subset of the data.

The proposed method for Class Representatives Selection (CRS) approximates the topological structure of the data by creating a reverse k-NN graph. CRS then iteratively selects nodes with the biggest reverse neighbourhoods as representatives of the data. This approach systematically minimizes the number of pairwise comparisons to reduce computational complexity while accurately representing the data.

3 Problem Formulation

In this section, we define the problem of prototype-based representation of classes and the nearest prototype classification (NPC). As we already stated in Introduction, we study the prototypes selection in general cases, including non-metric spaces. Therefore, we further assume that a class prototype is always specified as (possibly small) subset of its members.

Class Prototypes. Let T be an arbitrary space of objects for which a pairwise similarity function $s : \mathrm{T} \times \mathrm{T} \to \mathbb{R}$ is defined and let $X \subseteq \mathrm{T}$ be a set of (training) samples. Let $\mathcal{C} = \{C_1, ..., C_m\}$ be a set of classes of X such that $C_i \cap C_j = \emptyset, \forall i \neq j$ and $\bigcup C_i = X$. Let $C_i = \{x_1, x_2, ..., x_n\}$ be a class of size n. For $x \in C_i$, let

us denote U_x^k the k closest samples to x, i.e., the set of k samples that have the highest similarity to x in the rest of the class $C_i \setminus \{x\}$. Then the goal of the prototype selection is to find a prototype of class C_i, $R_i \subseteq C_i$ for each class such that:

$$\forall x \in C_i \; \exists \, r \in R_i \; : \; x \in U_r^k \tag{1}$$

In order to minimize computational requirements of NPC, we search for a minimal set of class representatives R_i^* for each class, which satisfies the coverage requirement (1):

$$R_i^* = \underset{|R_i|}{\arg\min} \left\{ r : \bigcup_{r \in R_i} U_r^k = C_i \right\} \tag{2}$$

Note that several sets might satisfy this coverage requirement.

Relaxed Prototypes. Finding class prototypes that fully meet the coverage requirement (1) might pose a computational burden and produce unnecessarily big prototypes. In most cases, covering the majority of the class objects while leaving out a few outliers leads to a smaller prototype that still captures the essential characteristics of a class. Motivated by this observation, we introduce a relaxed requirement for class prototypes. We say that a set $R_i \subseteq C_i$ is a representative prototype of class C_i if the following condition is met:

$$\left| \bigcup_{r \in R_i} U_r^k \cap C_i \right| \geq \epsilon \, |C_i|, \tag{3}$$

for a preset parameter $\epsilon \in (0, 1]$.

In further work, we replace the requirement (1) with its relaxed version (3) with $\epsilon = 0.95$. In case of need, the full coverage can be enforced by simply setting $\epsilon = 1$. Even in the relaxed version, we seek a prototype with minimal cardinality which satisfies (3).

Nearest Prototype Classification. Having the prototypes for all classes $\mathcal{R} = \{R_1, ..., R_m\}$, an unseen sample x is classified to the class with the most similar prototype $R^* \in \mathcal{R}$. R^* is the prototype containing representative r with the highest similarity to x.

$$r^* = \underset{r \in \bigcup R_i}{\arg\max} \, s(x, r).$$

Note that in our research we take into account only the closest representative $r*$. This choice comes from previous research [8] where 1-NN was yielded the best results.

4 Class Representatives Selection

In this section, we describe our method CRS for building the class prototypes. The entire method is composed of two steps:

1. Given a class C and a similarity measure s, a reverse k-NN graph G is constructed from objects C using the pairwise similarity s.
2. Graph G is used to select the representatives that satisfy the coverage requirement while minimizing the size of the class prototype.

The simplified scheme of the whole process is depicted in Fig. 1.

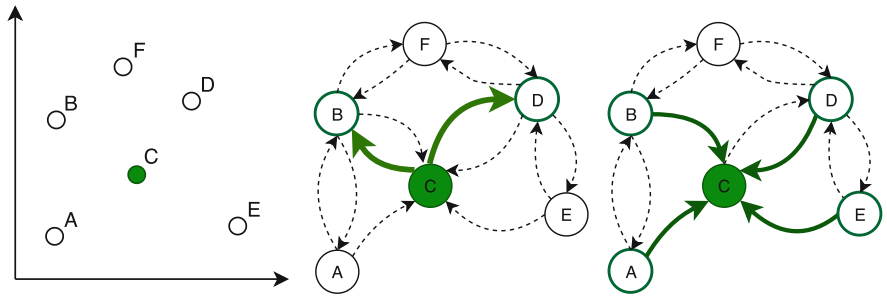


Fig. 1. Illustration of the steps of CRS algorithm. (a) Visualization of a toy 2D class. (b) 2-NN graph created from the class. (c) Reverse graph created from the graph depicted in (b). Node C's reverse neighbourhood covers A, B, D, E and thus would be a good first choice for a representative. Depending on the coverage parameter ϵ, the node F could be considered an outlier or also added to the representation

4.1 Building the Prototype

For the purpose of building the prototype for a class C a weighted reverse k-NN graph G_C^{-1} is used. It is defined as $G_C^{-1} = (V, E, w)$, where V is the set of all objects in the class C, E is a set of edges and w is a weight vector. An edge between two nodes $v_i, v_j \in V_{i \neq j}$ exists if $v_i \in U_{v_j}^k$, while the edge weight w_{ij} is given by the similarity s between the connected nodes, $w_{ij} = s(v_i, v_j)$.

The effective construction of such graph is enabled by employing the NN-Descent [18] algorithm, a fast converging approximate method for the k-NN graph construction. It exploits the idea that "a neighbour of a neighbour is also likely to be a neighbour" to locally explore neighbouring nodes for better solutions. NN-Descent produces a k-NN graph G_C. The reverse k-NN graph G_C^{-1} is then obtained from G_C by simply reversing directions of the edges in G_C.

Omitting all edges with weight lower than τ from the reverse k-NN graph G_C^{-1} ensures that very dissimilar objects do not appear in the reverse neighborhoods:

$$(\forall y \in U_x : s(x, y) \geq \tau)$$

The selection of representatives is treated as a minimum vertex cover problem on G_C^{-1} with omitted low similarity edges. We use a greedy algorithm which

Algorithm 1: Pseudocode for Class Representatives Selection

Data: class $C = \{c_1, ..., c_n\}$, similarity s, coverage threshold ϵ, size of neighbourhood k, weight threshold τ
Result: set of selected representatives $R \subseteq C$

G_C = NN-Descent(C, s, k)
G_C^{-1} = ReverseGraph(G_C, τ)
$Z = C$ *//set of uncovered objects*
$R = \{\}$ *//set of representatives*
while $\frac{|C|-|Z|}{|C|} < \epsilon$ **do**
  $r = \arg\max_c(|U_c|, c \in U_c)$
  $Z = Z \setminus U_r$
  $R = R \cup \{r\}$
end
return R

iteratively selects objects with maximal $|U|$ as representatives and marks them and their neighbourhood as covered. The algorithm stops when the coverage requirement (3) is met.

The whole algorithm is summarized in Algorithm 1.

4.2 Parameter Analysis

This subsection summarizes the parameters of the CRS method.

- k: number of neighbours for the k-NN graph creation. When k is high, each object covers more neighbours, but on the other hand it also increases the number of pairwise similarity calculations. This trade-off is illustrated for different values of k in Fig. 2. Due to the large impact of this parameter on properties of the produced representations and computational requirements, we further study its behaviour in more detail in a dedicated experiment in Sect. 5.
- ϵ: coverage parameter for the relaxed coverage requirement as introduced in Sect. 3. In this work, we set it to 0.95 which is a common threshold in outlier detection. It ensures that the vast majority of each class is still covered but outliers do not influence the prototypes.
- τ: threshold on weights, edges with lower weights (similarities) are pruned from the reverse neighbourhood graph G_C^{-1} (see Sect. 4.1). By default it is automatically set to approximate *homogeneity* $h(C)$ of the class C defined as:

$$h(C) = \frac{1}{|C|} \sum_{x_i, x_j \in C, i \neq j} s(x_i, x_j) \tag{4}$$

Additionally, the NN-Descent algorithm, used within the CRS method, has two more parameters that specify its behaviour during the k-NN graph creation. First, the δ_{nn} parameter which is used for early termination of the NN-Descent

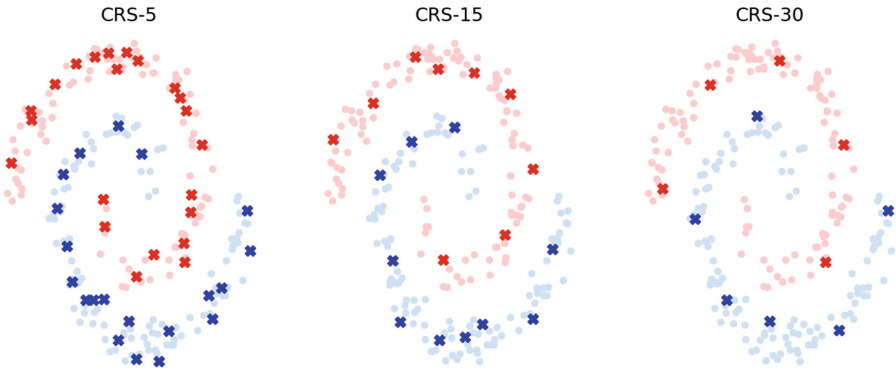


Fig. 2. In CRS the number of selected representatives and the quality of representation are both determined by k. For low ks the NN-Descent subsamples dense areas of the class too much and the information about neighbours is not propagated (CRS-5). As each object explores a bigger neighbourhood for higher k, the number of other objects it represents grows, therefore the number of representatives decreases. On the other hand, with less representatives, some information about the structure is lost, as in the case of $k = 30$

algorithm when the number of changes in the constructed graph is minimal. We set it to 0.001, as suggested by the authors of the original work [18]. Second, the sample rate ρ controls the number of reverse neighbours to be explored in each iteration of NN-Descent. Again, we set it to 0.5 to speed up the k-NN creation while not diverging too far from the optimal solution.

5 Experiments

This section presents experimental evaluation of the CRS algorithm on multiple datasets from very different domains that cover computer networks, text documents processing and image classification. First, we compare the CRS method to the state of the art techniques DS3 [9] and δ-medoids [10] on the nearest prototype classification task on different datasets. Then, we study the influence of the parameter k (which determines the number of nearest neighbours used for building the underlying k-NN graph).

We set δ in the δ-Medoids algorithm as approximate homogeneity h (see Eq. 4) calculated from random 5% of the class. Setting δ is a difficult problem not explained well in the original paper. From our experiments, homogeneity is a good estimate. The best results for DS3 we obtained with $p = \inf$ and $\alpha = 3$, while creating the full similarity matrix for the entire class. We tried $\alpha = 0.5$ which was suggested by the authors, but the algorithm always selected only one representative with much worse results. Finally, for CRS we set $\epsilon = 0.95$, $\tau = h$ (to be fair in comparison with δ-medoids). By far the most impactrul parameter is k. Section 5.4 looks at the selection in depth. A good initial choice for classes with 1000 or more samples is $k = 20$ and $k = 10$ works well for smaller classes.

5.1 Datasets

In this section we briefly describe the three datasets used in the following subsections for experimental comparison of individual methods.

MNIST Fashion. The MNIST Fashion [19] is a well established dataset for image recognition consisting of 60000 black and white images of fashion items belonging to 10 classes. It replaced the overused handwritten digits datasets in many benchmarks. Each image is represented by a 784 dimensional vector. In case of this dataset, the cosine similarity was used as the similarity function s.

20Newsgroup. 20Newsgroup dataset is a benchmark dataset for text documents processing. It is composed of nearly 20 thousand newspaper documents from 20 different classes (topics). The dataset was preprocessed such that each document is represented by a TF-IDF frequency vector of dimension 130,107. We used the cosine similarity which is a common choice in the domain of text documents processing as a similarity function s.

Private Network Dataset. Network dataset is the main motivation for our research. It was collected on a corporate computer network, originally for the purpose of network host clustering based on their behaviour [8]. The work defines a specific pair-wise similarity measure for network devices based on visited network hosts which we adopt for this paper. The dataset consists of all network communication collected on more than 5000 network hosts for one day (288 5-minute windows). This dataset resides in the space of all possible hostname and port number combinations. The dimension of this space is theoretically infinite, hence we work with a similarity that treats this space as non-metric.

For the purposes of the evaluation, classes smaller than 10 members were not considered, since such small classes can be easily represented by any method. The sizes and homogeneities of the classes can be found in Table 2. In contrast to the previous datasets, the sizes and values of homogeneity of classes in the Network dataset differ significantly, as can be seen in Table 2.

5.2 Evaluation of Results

In this section we present the results for each dataset in detail. The main results are summarized in Table 1. For a more complete picture we also included results for selecting a random 5% and all 100% of the class as a prototype. When evaluating the experiments, we take into account both precision/recall of nearest prototype classification and the percentage of samples selected as prototypes. Each method was run 10 times over a 80%/20% train/test split of each dataset. The results were averaged and the standard deviations of precisions and recalls were smaller than 0.005 for all methods, which shows stability of all algorithms. The only exception was δ-medoids on Network dataset where the precisions fluctuated up to 0.015.

Table 1. Average precision/recall values for each method used on each dataset. The table also shows the percentage of the class that was selected as a prototype. CRS outperforms both DS3 and δ-medoids on all datasets. In the network dataset CRS-k10 outperforms event the full-100% baseline as CRS does not try to cover outliers (in this case network hosts being very different from the rest of the class)

Method	MNIST Fashion	20Newsgroup	Network
δ-medoids	0.765/0.737 (10.45%)	0.362/0.3 (7.57%)	0.992/0.952 (7.01%)
DS3	0.727/0.715 (0.19%)	0.291/0.291 (0.57%)	0.853/0.956 (3.04%)
random-5%	0.780/0.767 (5.0%)	0.33/0.435 (5.0%)	0.94/0.959 (5.0%)
full-100%	0.849/0.846 (100.0%)	0.56/0.548 (100.0%)	0.987/0.963 (100.0%)
CRS-k10	0.823/0.817 (11.94%)	0.45/0.391 (11.28%)	0.992/0.973 (6.53%)
CRS-k20	0.813/0.806 (8.56%)	0.377/0.329 (6.58%)	0.972/0.976 (3.21%)
CRS-k30	0.806/0.798 (6.1%)	0.344/0.295 (4.83%)	0.983/0.968 (2.38%)

Table 2. Sizes and homogeneity for each class from network dataset. Classes with size lower than 10 were removed from the dataset

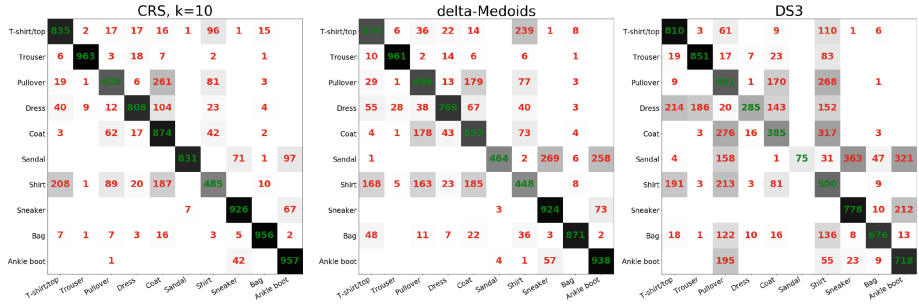
Class	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Size	1079	2407	75	2219	346	59	248	49	52	108	218	44	42	32
Homogeneity	0.58	0.14	0.84	0.64	0.60	0.92	0.34	0.84	0.69	0.35	0.78	0.35	0.79	1.0

As we have shown in the experiment in Sect. 5.4, CRS can be tuned by the parameter k to significantly reduce the number of representatives and maintain a high precision/recall values. The DS3 method selects a significantly lower number of representatives than any other method. However, it is at the cost of lower precision and recall values.

MNIST Fashion. The average homogeneity of a class in the MNIST Fashion dataset is 0.76. This corresponds with a slower decline of the precision and recall values as the number of representatives decreases. In Fig. 3 are the confusion matrices for the methods.

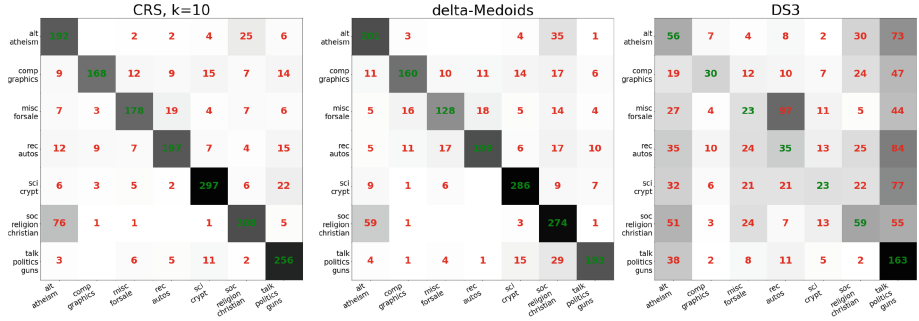
20Newsgroup. The 20Newsgroup dataset has the lowest average homogeneity $h = 0.0858$ from all the datasets. The samples are less similar on average, therefore the lower precision and recall values. Still CRS-k10 with only 11% of representatives performs quite well, compared with the other methods. Confusion matrices for one class form each subgroup are in Fig. 4.

Network Dataset. The results for data collected in real network further prove that lowering K does not lead a great decrease in performance. Again Fig. 5 shows confusion matrices for main 3 algorithms. Particularly interesting are the biggest classes A, B and D which were most difficult to cover for all algorithms.



(a) CRS-k10 (b) δ-Medoids (c) DS3

Fig. 3. Confusion matrices for each class in the MNIST Fashion dataset show the performance all 3 methods compared. The Sandal class was the hardest to represent for all methods



(a) CRS-k10 (b) δ-Medoids (c) DS3

Fig. 4. Confusion matrices for each class in the 20Newsgroup dataset show the performance all 3 methods compared

For sizes of all classes see Table 2. Moreover, lower homogeneity for B is also clearly seen in the confusion matrix.

5.3 Time Efficiency

When considering the speed of the algorithms, we particularly focus on cases where the slow and expensive computation of the pairwise similarity overshadows the rest, e.g. in the case of the Network dataset. Therefore, we compare the algorithms by the relative number of similarity computations S defined as:

$$S = \frac{S_{actual}}{S_{full}}, \tag{5}$$

where S_{actual} stands for the number of comparisons made and S_{full} is the number of comparisons needed for computing the full similarity matrix.

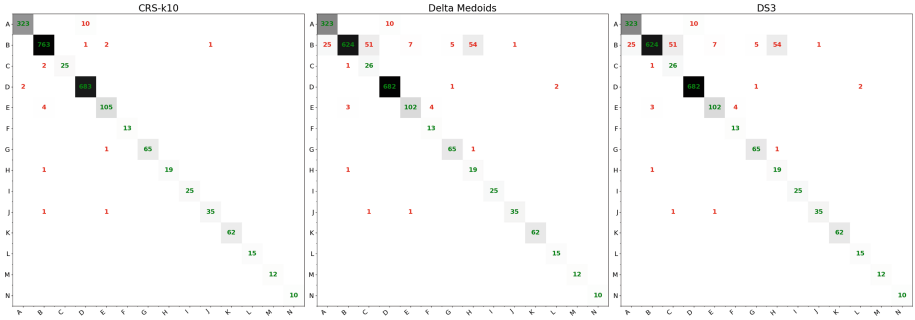


Fig. 5. Confusion matrices for each class in the Network dataset. Particularly interesting are the biggest classes A, B and D which were most difficult to cover for all algorithms. Moreover, lower homogeneity for B is also clearly seen in the confusion matrix

Table 3. The average number of similarity calculations relative to computing full similarity matrix in classes that have more than 1000 samples. For the DS3 algorithm, we always calculate the full similarity matrix; therefore, it is not included in the table

dataset	δ-Medoids		CRS-k10		CRS-k20	
	mean	std	mean	std	mean	std
MNIST Fashion	0.132	0.031	0.074	0.008	0.218	0.013
Network Dataset	0.467	0.266	0.178	0.058	0.507	0.16

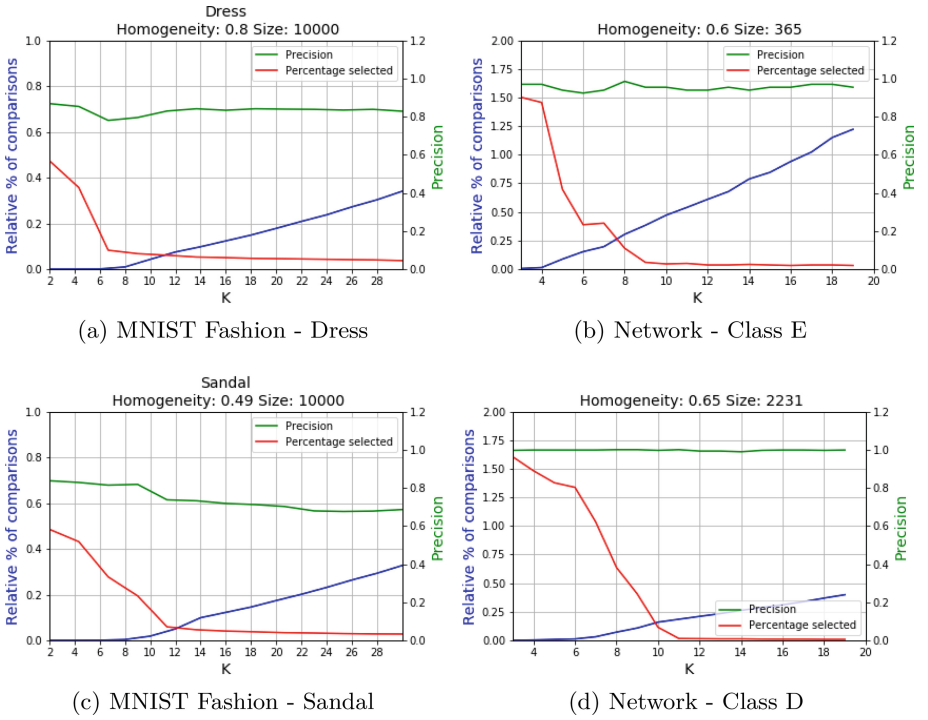
We measured S for classes bigger than 1000 samples to see how the algorithms perform on big classes. In smaller classes the total differences in comparisons are not great as the full similarity matrices are smaller. Also the smaller the class, the closer are all algorithms to $S = 1$ (for CRS it can be seen in Fig. 6h). The results for big classes are in Table 3. We use DS3 with the full similarity matrix to get most accurate results, therefore $S_{DS3} = 1$.

For CRS the number of comparisons is influenced by k, sample rate ρ, and homogeneity of each class and its size. However, we use very high ρ in the NN-Descent part of CRS, which significantly increases the number of comparisons. The impact of k is discussed in detail in Sect. 5.4 and experimenting with ρ is up for further research. For δ-Medoids the number of similarity computations performed is determined by the difficulty of the class and the δ parameter. In CRS, the parameters can be set according to the similarity computations we have available to achieve the best prototypes given the time. This does not hold neither for δ-medoids nor for $DS3$.

5.4 Impact of k

When building class prototypes by the CRS method, the number of nearest neighbours (specified by the parameter k) considered for building the k-NN

graph plays crucial role. With small ks, each object neighbours only few objects that are most similar to it. This also propagates into reverse neighbourhood graphs, especially for sparse datasets. Therefore, small ks increase the number of representatives needed to sufficiently cover the class. Using higher values of k produce smaller prototypes as each representative is able to cover more objects. The cost of this improvement is increased computational burden because the cost of k-NN creation increases rapidly with higher ks.



(a) MNIST Fashion - Dress

(b) Network - Class E

(c) MNIST Fashion - Sandal

(d) Network - Class D

Fig. 6. Illustration of how the selection of k influences the number of representatives and number of similarity computations. The number of representatives is in relative numbers to the size of the class. For different classes as k increases the relative number of comparisons also increases. However, the size of prototype selected decreases steeply while the precision decreases slowly (Color figure online)

Figure 6 shows trends of precision, sizes of created prototypes and numbers of similarity function evaluations depending on k for several classes that differ in their homogeneity and sizes. We can see the trade-off between computational requirements (blue line) and memory requirements (red line) as the k increases. From some point (e.g. where red line crosses the blue line), the classification precision decreases slowly. The cost limitations of building the prototype or the classification can be used to set the parameter k. If k is low, CRS selects prototypes faster, but the number of selected representatives is higher and therefore

the classification cost is also higher. If the classification cost (number of similarity computations to classify an object) is more important than prototype selection, parameter k can be higher.

6 Conclusion

This paper proposes CRS, a novel method for building representations of classes, class prototypes, which are small subsets of the original classes. CRS leverages nearest neighbour graphs to map each structure of each class and identify representatives that will form the class prototype. This approach allows CRS to be applied in any space where at least pairwise similarity is defined.

The proposed method was compared to the prior art in a nearest prototype classification setup on multiple datasets from different domains. The experimental results show that the CRS method achieves superior classification quality while producing comparably compact representations of classes.

References

1. Seo, S., Bode, M., Obermayer, K.: Soft nearest prototype classification. IEEE Trans. Neural Netw. **14**(2), 390–398 (2003)
2. Schleif, F.-M., Villmann, T., Hammer, B.: Local metric adaptation for soft nearest prototype classification to classify proteomic data. In: Bloch, I., Petrosino, A., Tettamanzi, A.G.B. (eds.) WILF 2005. LNCS (LNAI), vol. 3849, pp. 290–296. Springer, Heidelberg (2006). https://doi.org/10.1007/11676935_36
3. Cervantes, A., Galván, I., Isasi, P.: An adaptive michigan approach PSO for nearest prototype classification. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2007. LNCS, vol. 4528, pp. 287–296. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73055-2_31
4. Anthony, M., Ratsaby, J.: Large width nearest prototype classification on general distance spaces. Theoret. Comput. Sci. **738**, 65–79 (2018)
5. Martino, A., Giuliani, A., Rizzi, A.: Granular computing techniques for bioinformatics pattern recognition problems in non-metric spaces. In: Pedrycz, W., Chen, S.-M. (eds.) Computational Intelligence for Pattern Recognition. SCI, vol. 777, pp. 53–81. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89629-8_3
6. Becker, G.C.: Methods and apparatus for clustering templates in non-metric similarity spaces, October 12, US Patent 7,813,531 (2010)
7. Scheirer, W.J., Wilber, M.J., Eckmann, M., Boult, T.E.: Good recognition is non-metric. Pattern Recogn. **47**(8), 2721–2731 (2014)
8. Kopp, M., Grill, M., Kohout, J.: Community-based anomaly detection. In: 2018 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 1–6. IEEE (20180
9. Elhamifar, E., Sapiro, G., Sastry. S.S.: Dissimilarity-based sparse subset selection. IEEE Trans. Pattern Anal. Mach. Intell. **38**(11), 2182–2197 (2015)
10. Liebman, E., Chor, B., Stone, P.: Representative selection in nonmetric datasets. Appl. Artif. Intell. **29**(8), 807–838 (2015)
11. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Trans. Syst. Man Cybern. Part C (Appli. Rev.) **42**(1), 86–100 (2011)

12. Geva, S., Sitte, J.: Adaptive nearest neighbor pattern classification. IEEE Trans. on Neural Networks **2**, 2 (1991)
13. Xie, Q., Laszlo, C.A., Ward, R.K.: Vector quantization technique for nonparametric classifier design. IEEE Trans. Pattern Anal. Mach. Intell. **15**(12), 1326–1330 (1993)
14. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 417–435 (2012)
15. Wang, H., Kawahara, Y., Weng, C., Yuan, J.: Representative selection with structured sparsity. Pattern Recogn. **63**, 268–278 (2017)
16. Zhang, X., Zhu, Z., Zhao, Y., Chang, D., Liu, J.: Seeing all from a few: l1-norm-induced discriminative prototype selection. IEEE Trans. Neural Netw. Learn. Syst. **30**(7), 1954–1966 (2018)
17. Olvera-López, J.A., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.: Accurate and fast prototype selection based on the notion of relevant and border prototypes. J. Intell. Fuzzy Syst. **34**(5), 2923–2934 (2018)
18. Dong, W., Moses, C., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proceedings of the 20th International Conference on World Wide Web, pp. 577–586 (2011)
19. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)