Condensed Nearest Neighbour Rules for Multi-Label Datasets

PANAGIOTIS FILIPPAKIS[*] and STEFANOS OUGIAROGLOU, International Hellenic University, Greece
GEORGIOS EVANGELIDIS, University of Macedonia, Greece

Reducing the size of the training set, that is, replacing it with a condensing set, while maintaining the classification accuracy as much as possible is a very common practice to speed up instance-based classifiers. Data reduction techniques, also known as prototype selection or generation algorithms, can be used to accomplish this. There are numerous such algorithms that can be found in the literature that are effective for single-label classification problems, but the majority of them cannot be used for multi-label data where an instance may belong to multiple classes. Due to the numerous binary condensing sets it creates, the well-known Binary Relevance transformation method cannot be combined with a Data Reduction algorithm. Condensed Nearest Neighbor is a well-known parameter-free single-label prototype selection algorithm. This study proposes three variations of that algorithm for training datasets with multiple labels. An experimental study that we conducted over nine distinct datasets shows that our three proposed approaches provide good reduction rates while not tampering with the classification rates.

CCS Concepts: • **Information systems → Nearest-neighbor search**; • **Computing methodologies → Instance-based learning**.

Additional Key Words and Phrases: data reduction techniques, instance reduction, multi-label classification, prototype selection, instance-based classification, binary relevance, CNN, BRkNN

ACM Reference Format:
Panagiotis Filippakis, Stefanos Ougiaroglou, and Georgios Evangelidis. 2023. Condensed Nearest Neighbour Rules for Multi-Label Datasets. In . ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3589462.3589492

1 INTRODUCTION

In contrast to single-label classification problems where an instance belongs to one label, a multi-label classification task [14] refers to the process of classifying an instance into multiple classes. This approach is commonly used for classifying various types of data, including images, books, artists, music, videos and movies. For example, a movie can be classified as both “crime" and “adventure", a text can concern politics and sports, a music track may belong to multiple genres or moods, and an image may depict “mountain", “sea", and “beach" simultaneously. The multi-label classification problem is a generalization of the single-label problem, and provides a more comprehensive understanding of the classification process.

The *k*-Nearest Neighbors (*k*-NN) classification algorithm is a common example of a lazy or instance-based classifier, as it retrieves the *k* nearest neighbors of an unclassified instance and applies a majority voting method to classify it. In other words, the unclassified instance is classified to the most common class among the classes of the retrieved *k* nearest neighbours. This classifier is known for its simplicity, ease of implementation, and strong classification performance, making it a useful tool for both single-label and multi-label classification tasks. However, it involves high computational cost because it needs to figure out how far each instance that must be classified is from each instance in the training set.

[*]All authors contributed equally to this research.

Therefore, the size of the training set in instance-based classification is a crucial issue. A large training set results in high classification accuracy but also in high computational cost. To speed-up the k-NN classifier, it is often necessary to reduce its memory and CPU requirements by reducing the size of the training set. One way to accomplish this in single-label classification tasks is to use a Data Reduction Technique (DRT) that is able to reduce either the training instances or the number of attributes [7]. This paper considers DRTs from instance reduction point of view. The goal of this paper is to accomplish fast k-NN classification on multi-label data by reducing the size of the training set without harming accuracy.

DRTs can be either Prototype Selection (PS) [4] or Prototype Generation (PG) [13]. In practice, they are data pre-processing tasks that replace the initial training dataset with a smaller subset called the "condensing set". By using the condensing set, the k-NN classifier can achieve as high accuracy as that of using the original training dataset but with significantly lower computational cost. Prototype selection algorithms select instances, or prototypes, from the original training set, while prototype generation algorithms generate prototypes by summarizing similar training instances from the same class. The basic idea behind many DRTs is that only training instances close to the class decision boundaries in an Euclidean metric space are necessary for classification tasks. Those training instances that belong to the "internal" area of a class, far from the decision boundaries, can be safely removed without sacrificing classification accuracy. Therefore, DRTs aim to select or generate enough prototypes that are near the decision boundary data areas for each class. The vast majority of DRTs concern single-label classification problems.

The technique known as the Label Powerset (LP) transformation [14] can provide a simple solution for using a DRT in a multi-label classification problem. LP transforms a multi-label dataset into a single-label dataset by treating each label combination (or labelset) as a distinct class. However, it is important to note that LP is only suitable when the number of labels and possible labelsets is small and there are enough instances for each labelset. In cases where the number of label combinations is too large, the reduction rate may be insufficient, and some combinations may be poorly represented. Additionally, the total number of different label combinations may increase exponentially, which can lead to scalability issues.

Binary Relevance (BR) is another one well-known transformation technique that transforms a multi-label classification problem to a single label classification problem. In effect, BR involves converting the multi-label problem into several single-label binary problems. To predict the labels that an unclassified instance belongs to, one needs as many classifiers as there are available labels. The combination of BR with the k-NN classifier is referred to as BRkNN [12]. The combination is effective because the k-NN classifier is a lazy classifier that does not construct any classification model. When classifying an instance x, BRkNN searches for the k nearest neighbours to x just like the single-label k-NN does. Next, the nearest neighbours' voting procedure is repeated once for each label.

The k-NN classifier becomes eager when a DRT is used beforehand. In such cases, the classification model is the condensing set. However, when dealing with multi-label classification, constructing a condensing set for each label using BR causes the data reduction goal to fail. As a result, the k-NN classifier must search for nearest neighbors in each condensing set to make individual label predictions. Therefore, it is not possible to combine BR with a DRT due to the existence of multiple binary condensing sets. Thus, it is clear that DRTs need to be modified to manage multi-label datasets. This constitutes the motivation behind the current work.

The Condensed Nearest Neighbour (CNN) rule [10] is the oldest and a state-of-the-art PS algorithm for single label classification problems that works by removing training instances that lie far for the decision boundaries. This paper attempts to develop variations of CNN for multi-label classification problems. Therefore, the contribution of the paper is the development of new multi-label training data reduction techniques based on CNN.

The rest of this paper is organized as follows. Section 2 presents the related work in data reduction on multi-label data-sets. Section 3 presents the CNN algorithm for single label classification problems. Section 4 describes the proposed variations of CNN for multi-label classification problems that use with the BR*k*NN multi-label classifier. Section 5 presents the experimental study that compares the proposed algorithms. Finally, Section 6 concludes the paper.

2 RELATED WORK

Most publications that deal with multi-label problems refer to classification algorithms rather than techniques that will reduce the computational costs of large multi-label training sets. Few papers deal with data reduction for these kinds of datasets. In this section, we review the scant relevant literature.

The authors of [3] propose a PS algorithm for multi-label datasets. The algorithm eliminates noise during editing and balances the class decision boundaries. The authors, who were motivated by the Edited Nearest Neighbour rule (ENN-rule) [16], suggest an under-sampling method for training sets that are imbalanced.

In article [5], the authors propose a prototype selection editing algorithm based on ENN-rule. The Metric of Hamming loss is used by the algorithm to identify the noisy training instances. The concept is straightforward: The instances with high Hamming loss should be eliminated because they are likely to be close to decision boundaries, much like the ENN-rule.

The article [6] is the first attempt to adapt PS algorithms to multi-label problems. The proposed algorithms are based on local sets [1] and on the LP transformation technique [14]. In single-label problems, the local set of an instance x is the largest set of instances centered on x that are all members of the same class. The authors claim that in multi-label datasets, it is not necessary for a local set to contain instances with the exact same labelset. A local set's instances might have marginally different labelsets. The authors compute the Hamming loss over the labelsets to measure the differences between labelsets. The distance among two instances is measured using the Hamming loss of their labelsets. If the distance is greater than a set threshold, the instances are classified as belonging to different "classes".

In article [17], the authors use single-label prototype selection algorithms along with Binary Relevance (BR), Label Powerset (LP), and other transformation techniques. For BR and its variants, the proposed strategy generates as many single-label training sets as the different label types. Then, each training set is subjected to a prototype selection algorithm to produce a condensing set for each label. A vote is cast in favor of an instance each time it is selected, and all of the votes for all instances are added up to form a single vector. By choosing all instances with more votes than a predetermined threshold, the strategy creates a complete condensing set.

Article [8] is an attempt to use homogeneous clustering to reduce multi-label datasets. The proposed algorithm is called Multi-label Reduction through Homogeneous Clusters (MLRHC) and is an adaptation of the single label prototype generation algorithm RHC, which repeatedly applies K-means clustering to produce homogeneous clusters that are replaced by their center. For MLRHC, a cluster is considered homogeneous when all instances in the cluster have at least one common label. MLRHC performs K-means clustering on the initial dataset using the representatives of existing labels in the dataset as the initial means and finds as many clusters as the existing labels. Homogeneous clusters are replaced by their center that is labeled by the common label along with each label appearing at least in half of the cluster's instances. In a similar manner, K-means is applied on the non-homogeneous clusters until all clusters become homogeneous. In the same article, the authors propose an RSP3 variant called Multi-label Reduction by Space Partitioning (MLRSP3) that also makes use of the idea that a cluster is homogeneous if its instances have at least one common label. Assuming that the initial training set is a non-homogeneous cluster, MLRSP3 retrieves the two farthest instances in the training set. The training set is then divided into two clusters by assigning its instances to the closest farthest instance. In a similar manner, MLRSP3 divides all non-homogeneous clusters until all clusters become homogeneous. Like MLRHC, the center of homogeneous clusters becomes a multi-label prototype labeled by the common label and all labels that appear in at least half the instances of the cluster.

3 THE SINGLE LABEL CONDENSED NEAREST NEIGHBOUR RULE

As mentioned in Section 1, the Condensed Nearest Neighbour (CNN) rule [10] is the first and the most widely used prototype selection algorithm. It is a parameter-free single-label prototype selection algorithm that builds its condensing set by scanning the training data multiple times.

The CNN method uses two store areas, which we will refer to as Condensing set (CS) and Training set (TS). Initially, TS contains the entire training set and CS is empty. One instance is randomly chosen from TS and transferred to CS to start the process. Each instance $x \in$ TS is then compared to those that are currently stored in CS.

More specifically, for each instance $x \in$ TS, CNN finds its nearest neighbour (or 1-NN) in the current CS using Euclidean distance. If x is correctly classified by its Nearest Neighbour in CS, it is kept in TS. Otherwise, x is removed from TS and is added to CS. Once all $x \in$ TS have been taken into account, the process continues by a next scan of TS. The algorithm stops when all instances in TS are correctly classified by the content of CS. In other words, CNN terminates when no instance is transferred from TS to CS during a complete scan of TS. Algorithm 1 presents CNN is pseudo-code.

Algorithm 1 CNN

```
                    Input: $TS$
                    Output: $CS$
1:  $CS \leftarrow \emptyset$
2:  randomly pick an instance of TS and move it to CS
3:  repeat
4:      $stop \leftarrow TRUE$
5:      for each $x \in TS$ do
6:          $NN \leftarrow$ nearest neighbour of $x$ in $CS$ using Euclidean distance
7:          if $NN_{class} \neq x_{class}$ then
8:              $CS \leftarrow CS \cup \{x\}$
9:              $TS \leftarrow TS - \{x\}$
10:             $stop \leftarrow FALSE$
11:         end if
12:     end for
13: until $stop == TRUE$ {no move during a pass of TS}
14: discard TS
15: return $CS$
```

The main principle of CNN is that instances that are incorrectly classified must be included in the Condensing Set (CS) because they are border instances, i.e, close to decision boundaries. CNN makes sure that every removed TS instance can be correctly classified using the information in the CS set. The fact that the CNN has no parameters is a significant advantage, but, there are some disadvantages, too:

- Multiple runs of the algorithm on a TS may produce different condensing datasets, in case a different initial randomly selected instance moves to CS (line 2 of Algorithm 1) or the TS instances are examined in a different order (line 5 of Algorithm 1).

- The algorithm is memory based. All the instances should reside in main memory.
- The algorithm makes multiple passes over the training set. Thus, it has high computational cost.

In terms of quality, CNN acts as follows: The algorithm tends to select instances close to the (possibly fuzzy) boundary between the classes if the Bayes risk is low, that is, if the underlying densities of the different classes have little overlap. Deeply ingrained instances will not typically be transferred to CS because they will be correctly classified. If the Bayes risk is high, CS will essentially contain every instance from the initial TS set, and no meaningful sample size reduction will be performed.

4 THE PROPOSED ALGORITHMS

As mentioned in Section 1, the "Conventional" data reduction algorithms are not suitable to be applied in conjunction with the Binary Relevance transformation method in multi-label data since this would result in the creation of many condensing sets, one for each label.

This section presents CNN variations that are appropriate for Multi-label datasets. The proposed algorithms are called Multi-label Condensed Nearest Neighbour 1 (MLCNN-1), Multi-label Condensed Nearest Neighbour 2 (we propose two versions of MLCNN-2), and, Multi-label Condensed Nearest Neighbour 3 (MLCNN-3).

4.1 MLCNN-1

The idea behind MLCNN-1 is similar to the one behind CNN: an instance that has a labelset that is quite different from that of its nearest neighbour should be placed in the multilabel condensing set.

Hence, MLCNN-1 needs a way to measure the difference (distance) between multi-label instances and a method to decide when two instances differ or are similar.

The labelset of an instance is a sequence of bits (0 or 1), where 0 denotes that the instance does not belong to the corresponding label and 1 denotes that the instance belongs to the label. To compute the Hamming loss (HL) given two instances, one performs an XOR operation on their labelsets to count how many labels differ. That number is divided by the length of a labelset, or in other words, the total number of labels in the dataset. Therefore, when HL equals zero the two instances have identical labelsets. On the other hand, when HL equals one the lablesets are completely different.

MLCNN-1 is based on the concept of the label density of the dataset. Label density of a dataset is the average number of labels of the instances of the dataset divided by the number of distinct labels in the daataset [2]. MLCNN-1 considers that the labelesets of the examined instances are quite different, when the HL between them is greater than the dataset density.

MLCNN-1 works similar to single label CNN. Thus, it uses two store sets: the Condensing set (CS) and the Training set (TS). Initially, TS contains the entire training set and CS is empty. MLCNN-1 selects a random instance from TS and transfers it to CS. For each instance $x \in$ TS, the algorithm finds the nearest neighbour (for example y) in the current CS. Then, MLCNN-1 calculates the Hamming loss metric (HL) between x and y. HL quantifies how different the labelsets of the two instances are. If HL is higher than the density of dataset, x is removed from TS and is added to CS. Otherwise, it remains in TS. Once all $x \in$ TS have been examined, the process continues with the next scan of the remaining instances in TS. MLCNN-1 stops when no move from TS to CS is performed during a complete pass of TS.

4.2 MLCNN-2

The second MLCNN variation uses the Jaccard Distance metric. Jaccard distance is a metric for measuring the dissimilarity of two sets. Its standard version is computed by taking the ratio of the difference between the set union and the set intersection over the set union.

In the case of labelsets, we use the version of Jaccard Distance of asymmetric binary attributes, i.e., we consider the presence of a label to be more important than its absence. Hence, the matching zeros are ignored from the computation of the distance between two labelsets. The Jaccard Distance of asymmetric binary attributes (JD) is computed as the ratio of the number of non matching labels over the number of matching appearing labels. Below, we give some examples:

- JD(110001, 110001) = 0/3 = 0
- JD(110001, 001110) = 6/6 = 1
- JD(110001, 000010) = 4/4 = 1
- JD(110001, 100010) = 3/4 = 0.75

MLCNN-2 is based on the idea of MLCNN-1: An instance with a labelset that is quite different from that of its nearest neighbour in CS should be also placed in CS as prototype. The difference between MLCNN-1 and MLCNN-2 is that the latter utilizes Jaccard distance instead of HL and that a different JD threshold is used to determine how much two instances differ. We expect the largest JD threshold to increase the reduction rate since more instances will be considered similar to their nearest neighbor and will not enter the CS. Initially, we consider as different only the instances that share fewer than half the labels (JD threshold equals 0.5). However, we examined higher threshold values in order to increase the reduction rates. After experimentation, we decided to choose and test two JD threshold values, namely, 0.5 and 0.75.

4.3 MLCNN-3

MLCNN-3 follows a completely different approach compared to MLCNN-1 and MLCNN-2. Initially, MLCNN-3 utilizes the Binary Relevance transformation method in order to transform the multilabel problem with |L| labels, into |L| single label problems. Then, MLCNN-3 applies CNN to each label of the training set separately. Through this process, the algorithm builds as many condensing sets as the number of labels of the original training set. For example, if the initial training set has five labels, five condensing sets are generated. Each condensing set contains prototypes of class 1 (the instance belongs to the corresponding label) and 0 (the instance does not belong to the corresponding label).

Then, MLCNN-3 selects from each condensing set only the prototypes that have the value 1 in their corresponding label. The prototypes for label 0 are discarded. In the end, MLCNN-3 merges all the multiple condensing sets and creates the final condensing set. Initially, all the selected prototypes of the first label are placed in the final multi-label condensing set. Then, each prototype x of the second label is examined. If the prototype already exists in the final condensing set, its label is added in the labelset of that prototype. Otherwise, x is added in the final multi-label condensing set and its labelset contains the second label. The process is repeated for all the remaining condensing sets.

To better explain the way MLCNN-3 works, we provide the following example. Let us assume that MLCNN-3 runs CNN for each label of an original two dimensional training dataset that has two labels. Suppose that the two condensing sets presented in Tables 1 and 2 are derived. The first condensing set contains SIX prototypes labeled by "1", which means that the prototype belongs to the corresponding label. The second condensing set contains four prototypes labeled by "1". The final multi-label condensing set constructed by the merging procedure of MLCNN-3 is shown in Table 3. It contains eight prototypes. Prototypes (1,1), (1,8), (4,5) and (9,1) originate exclusively from the condensing set of the first label, prototypes (5,6) and (9,9) originate exclusively from the condensing set of the second label, while prototypes (3,8) and (8,4) are common in both condensing sets.

Table 1. Condensing set for the First label.

Instances	First Label
(1,1)	1
(1,8)	1
(2,7)	0
(3,8)	1
(4,5)	1
(7,1)	0
(8,4)	1
(9,1)	1

Table 2. Condensing set for the Second label.

Instances	Second Label
(1,1)	0
(2,7)	0
(3,8)	1
(5,6)	1
(7,5)	0
(8,4)	1
(9,9)	1

Table 3. Final merged condensing set.

Instances	First merged Label	Second merged Label
(1,1)	1	0
(1,8)	1	0
(3,8)	1	1
(4,5)	1	0
(5,6)	0	1
(8,4)	1	1
(9,1)	1	0
(9,9)	0	1

5 EXPERIMENTAL STUDY

5.1 Experimental setup

In our experimentation, we used nine multilabel datasets distibuted by Mulan [15]. We used datasets with numeric features containing at least five hundred (500) instances. Table 4 summarizes the major features of the used datasets. Dataset cardinality and density are shown in the final two columns. The cardinality is the average number of labels per instance. Calculating the density involves dividing the cardinality by the total number of labels. The domain of each dataset is listed in the second column of Table 4.

The datasets contain features of different value ranges. This can affect the process of classification since high valued features will dominate the distance computation among instances. For this reason, we normalized the values of all features to the [0,1] range. The normalization was performed using the MinMaxScaler from scikit-learn Python library [9].

Then, all datasets were divided into training and test sets using the stratified 5-fold cross-validation method [11]. Stratified cross-validation reduces the variance of the estimates and improves the estimation of the generalization performance of classification algorithms. Stratified sampling ensures that the proportion of the feature of interest is the same in the original data, training set, and, test set. This guarantees that no value is over-represented or under - represented in the training and test sets, providing a more precise estimate of performance and error.

CNN and the proposed CNN variations were implemented using Python3 and Multiprocessing. Multiprocessing allows two or more CPU threads to process two or more distinct parts of the same python script at the same time, with the main benefit in speed and the ability to handle larger amounts of data.

Table 4. Dataset characteristics

Datasets	**Domain**	**Size**	**Attributes**	**Labels**	**Cardinality**	**Density**
CAL500 (CAL)	Music	502	68	174	26.044	0.150
Emotions (EMT)	Music	593	72	6	1.869	0.311
Water quality (WQ)	Chemistry	1060	16	14	5.073	0.362
Scene (SC)	Image	2407	294	6	1.074	0.179
Yeast (YS)	Biology	2417	103	14	4.237	0.303
Birds (BRD)	Sounds	645	260	19	1.014	0.053
CHD49 (CHD)	Medicine	555	49	6	2.580	0.430
Image (IMG)	Image	2000	294	5	1.236	0.247
Mediamill (MDM)	Video	43907	120	101	4.376	0.043

We compared the performance of BR*k*NN when applied on the condensing set created by MLCNN-1, the two versions of MLCNN-2 and MLCNN-3 with that of BR*k*NN when applied on the initial training set (without data reduction). Reduction rate and Hamming loss were obtained using a five-fold stratified cross-validation schema. Since the goal of the proposed variations is to speed-up instance-based classifiers in multi-label domains, the use of eager multi-label classifiers in the experimental study does not make sense.

Because the computational cost of the BR*k*NN classifier depends on the size of the training set used, the CPU time needed for classification is not reported. In effect, the higher the reduction rate is, the lower classification computational cost is required by BR*k*NN. The prediction effectiveness as measured by computing Hamming loss, which is the proportion of incorrectly predicted labels to all labels. Hamming loss is computed as follows:

$$HL = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \Delta Z_i|}{|L|}$$

Where Y_i is the set of actual labels for each instance and Z_i is the set of predicted labels for each instance, m is the overall number of instances in the dataset, and, $|L|$ is the total number of labels. The symmetric difference of two sets, or $Delta$, is what the XOR operation looks like. For example, if the actual labelset of a testing instance $x1$ is 11001 and the predicted labelset is 11010, then the Hamming loss is $\frac{2}{5} = 0.4$. Similarly, if the actual labelset of a testing instance $x2$ is 00001 and the predicted labelset is 11010, then the Hamming loss is $\frac{4}{5} = 0.8$. Then, the HL for a testing set consisting of those two instances, will be $\frac{1}{2} * (0.4 + 0.8) = 0.6$. Finally, following a common practice in the relevant literature (e.g. [4, 13]), all experiments were conducted using k=1.

Table 5. Comparison Table of the Reduction Rate (RR (%)) and the Hamming Loss (HL(%))

Dataset		**BRkNN** $k = 1$	**MLCNN-1 BRkNN**	**MLCNN-2 (JD>0.5) BRkNN**	**MLCNN-2 (JD>0.75) BRkNN**	**MLCNN-3 BRkNN**
CAL	HL:	0.19	0.19	0.19	0.19	0.19
	RR:	-	8.28	0.75	19.45	0.0
EMT	HL:	0.24	0.26	0.26	0.25	0.29
	RR:	-	39.88	40.26	60.62	26.64
WQ	HL:	0.33	0.34	0.33	0.34	0.36
	RR:	-	45.71	18.99	54.65	5.33
SC	HL:	0.11	0.12	0.12	0.12	0.13
	RR:	-	51.93	51.93	53.03	50.44
YS	HL:	0.24	0.27	0.26	0.26	0.30
	RR:	-	48.26	31.27	54.72	16.61
BRD	HL:	0.05	0.06	0.05	0.05	0.08
	RR:	-	55.70	15.50	22.71	58.18
CHD	HL:	0.35	0.36	0.36	0.38	0.40
	RR:	-	42.18	32.20	58.64	14.74
IMG	HL:	0.20	0.22	0.22	0.21	0.23
	RR:	-	42.11	42.11	44.82	38.43
MDM	HL:	0.03	0.03	0.03	0.03	0.04
	RR:	-	48.70	33.19	54.58	21.76

5.2 Experimental results

The results of the experimental study are shown in Table 5. We provide the Hamming loss and the reduction rate achieved by MLCNN-1, MLCNN-2 and MLCNN-3 for each dataset. We note that MLCNN-1 attained reduction rates ranging from 8.28% to 55.70%, MLCNN-2 (JD>0.5) reduction rates ranging from 0.75% to 51.93%, MLCNN-2 (JD>0.75) reduction rates ranging from 19.45% to 60.62%, and, MLCNN-3 reduction rates ranging from 0% to 58.18%.

Figure 1 demonstrates the decrease in the number of instances following the use of the MLCNN-1 prototype selection method. Figures 2 and 3 show the decrease in instances following the use of the two versions of the MLCNN-2 prototype selection method. Finally, Figure 4 shows the decrease in the number of instances following the use of the MLCNN-3 prototype selection method.

We observe that the highest reduction rate on average is achieved by MLCNN-2 (JD > 0.75), followed by MLCNN-2 (JD > 0.5) and MLCNN-3. In the end, an important factor that affects the reduction rate is the way that instances are distributed within the dataset.

Let's not forget that in essence we are dealing with complex data that do not have a constant distribution in space, which is why we observe large fluctuations in the reduction rate per data set.

In general, however, we observe that MLCNN-3 exhibits less stable behavior in terms of the reduction rate achieved per dataset than the rest of the algorithms.

Furthermore, we see that there is no difference in Hamming loss between the BR*k*NN classifier that uses the initial training set and the BR*k*NN classifier that uses the multi-label condensing sets created by MLCNN-1, MLCNN-2 (JD > 0.5), MLCNN-2 (JD > 0.75) and MLCNN-3. We can therefore conclude that the proposed algorithms significantly improve reduction rates while maintaining accuracy at high levels.

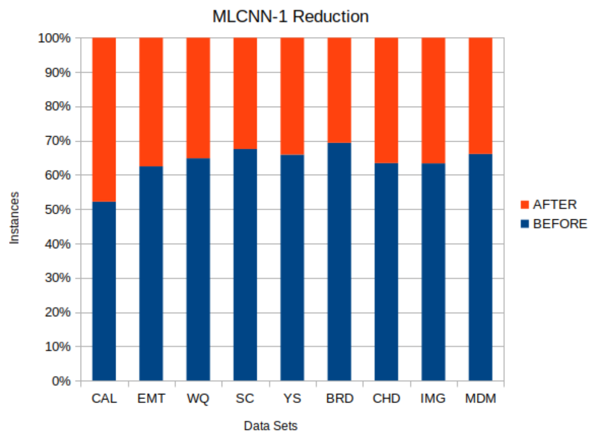

Fig. 1. Number of instances using MLCNN-1

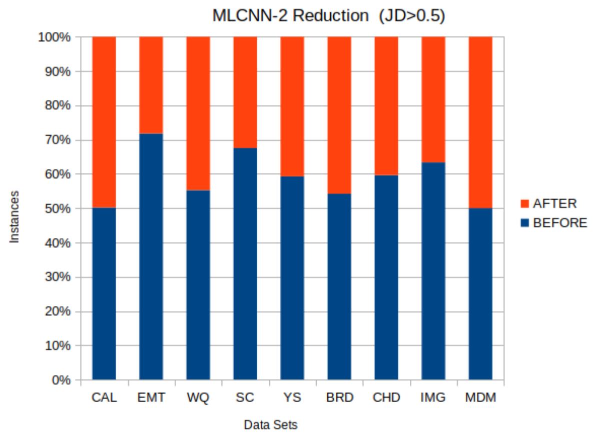

Fig. 2. Number of instances using MLCNN-2 (JD>0.5)

6 CONCLUSIONS

In this paper we deal with data reduction techniques for multi-label datasets. By data reduction we refer to instance reduction and not feature reduction. This type of reduction is an essential pre-processing step in instance-based classification in order to avoid the drawbacks of high computational cost.

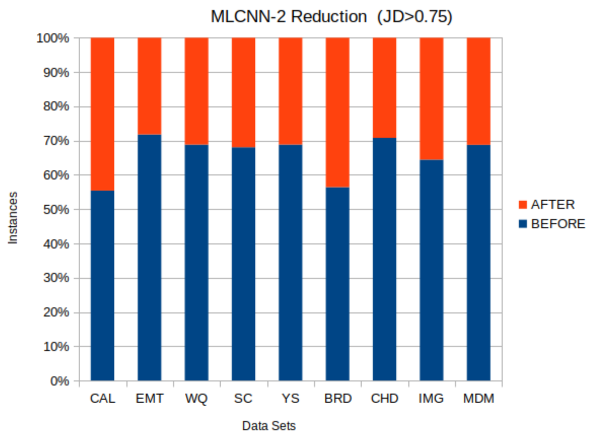

Fig. 3. Number of instances using MLCNN-2 (JD>0.75)

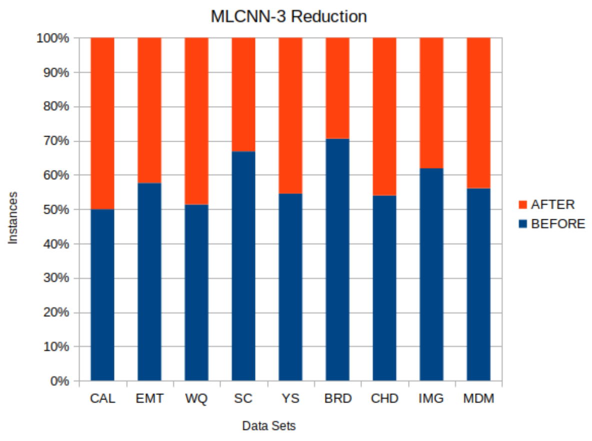

Fig. 4. Number of instances using MLCNN-3

However, the vast majority of data reduction techniques currently in use do not apply to multi-label classification problems and do not combine well with problem transformation methods, like Binary Relevance or Label Powerset.

In order to accelerate the k-NN classifier in the context of multi-label classification, recent research efforts were first presented in this paper. Then, three variations of the well-known Condensed Nearest Neighbor rule that are suitable for multi-label classification were proposed. The proposed MLCNN-1, MLCNN-2 (JD > 0.5), MLCNN-2 (JD > 0.75), and MLCNN-3 algorithms can be regarded as the first prototype Selection algorithms for data condensing.

All the proposed algorithms are parameter-free. MLCNN-1 considers two neighboring instances as different when their Hamming loss distance is greater than the dataset density. MLCNN-2 considers two neighboring instances as different when their Jaccard distance is greater than a given threshold. Finally, MLCNN-3, initially builds different prototypes for each label by applying the conventional CNN multiple times. Then, it merges the prototypes by combining the different labels and builds the final condensing sets.

As a result, MLCNN-1, both versions of MLCNN-2, and MLCNN-3 create a multi-label condensing set that BR*k*NN can use to perform multi-label prediction by finding nearest neighbors. The experimental study demonstrated that the accuracy attained by BR*k*NN was unaffected by switching from the initial training set to the condensing sets produced by MLCNN-1, the two versions of MLCNN-2, and MLCNN-3. However, the computational cost needed for the classification process is decreased when condensing sets are used. The new variations usually reduce the computational cost by more than 50%.

MLCNN-1 and the two versions of MLCNN-2 outperformed MLCNN-3 in terms of accuracy, while achieving higher reduction rates. Also, MLCNN-2 (JD > 0.75) outperforms the rest of the algorithms in terms of overall classification performance. This study demonstrated that data reduction on multi-label problems is still an active area of study in the context of data mining and machine learning. We intend to adapt popular single-label data reduction techniques on multi-label datasets. The next step is to develop novel, parameter-free data reduction and scalable, multi-label training set classification methods.

REFERENCES

[1] Henry Brighton and Chris Mellish. 2002. Advances in Instance Selection for Instance-Based Learning Algorithms. *Data Mining and Knowledge Discovery* 6, 2 (2002), 153–172. https://doi.org/10.1023/a:1014043630878

[2] Adam Byerly and Tatiana Kalganova. 2022. Class Density and Dataset Quality in High-Dimensional, Unstructured Data. https://doi.org/10.48550/arxiv.2202.03856

[3] Francisco Charte, Antonio J. Rivera, María J. del Jesus, and Francisco Herrera. 2014. MLeNN: A First Approach to Heuristic Multilabel Undersampling. In *Intelligent Data Engineering and Automated Learning – IDEAL 2014*. Springer International Publishing, New York, NY, USA, 1–9. https://doi.org/10.1007/978-3-319-10840-7_1

[4] Salvador Garcia, Joaquin Derrac, Jose Cano, and Francisco Herrera. 2012. Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. *IEEE Trans. Pattern Anal. Mach. Intell.* 34, 3 (March 2012), 417–435. https://doi.org/10.1109/TPAMI.2011.142

[5] Sawsan Kanj, Fahed Abdallah, Thierry Denœux, and Kifah Tout. 2015. Editing training data for multi-label classification with the k-nearest neighbor rule. *Pattern Analysis and Applications* 19, 1 (Feb 2015), 145–161. https://doi.org/10.1007/s10044-015-0452-8

[6] Enrique Leyva, Antonio González, and Raúl Pérez. 2015. Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognition* 48, 4 (2015), 1523–1537. https://doi.org/10.1016/j.patcog.2014.10.001

[7] Huan Liu and Hiroshi Motoda. 1998. *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic Publishers, USA.

[8] Stefanos Ougiaroglou, Panagiotis Filippakis, Georgia Fotiadou, and Georgios Evangelidis. 2023. Data reduction via multi-label prototype generation. *Neurocomputing* 526 (2023), 1–8. https://doi.org/10.1016/j.neucom.2023.01.004

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, 85 (2011), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

[10] P.E.Hart. 1967. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory* vol.18 (Jan. 1967), pp 515–516.

[11] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2011. On the Stratification of Multi-label Data. In *Machine Learning and Knowledge Discovery in Databases*, Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 145–158. https://doi.org/10.1007/978-3-642-23808-6_10

[12] E. Spyromitros, G. Tsoumakas, and Ioannis Vlahavas. 2008. An Empirical Study of Lazy Multilabel Classification Algorithms. In *Artificial Intelligence: Theories, Models and Applications*, John Darzentas, George A. Vouros, Spyros Vosinakis, and Argyris Arnellos (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 401–406. https://doi.org/10.1007/978-3-540-87881-0_40

[13] Isaac Triguero, Joaquín Derrac, Salvador Garcia, and Francisco Herrera. 2012. A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. *Trans. Sys. Man Cyber Part C* 42, 1 (Jan. 2012), 86–100. https://doi.org/10.1109/TSMCC.2010.2103939

[14] Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3, 3 (1 Jan. 2007), 1–13. https://doi.org/10.4018/jdwm.2007070101

[15] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. *Mining Multi-label Data.* Springer US, Boston, MA, 667–685. https://doi.org/10.1007/978-0-387-09823-4_34

[16] Dennis L. Wilson. 1972. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-2, 3 (July 1972), 408–421. https://doi.org/10.1109/tsmc.1972.4309137

[17] Álvar Arnaiz-González, José-Francisco Díez-Pastor, Juan J Rodríguez, and César García-Osorio. 2018. Local sets for multi-label instance selection. *Applied Soft Computing* 68 (2018), 651–666. https://doi.org/10.1016/j.asoc.2018.04.016

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009