Efficient Support Vector Machine Classification using Prototype Selection and Generation

Stefanos Ougiaroglou[1], Konstantinos I. Diamantaras[1], and Georgios Evangelidis[2]

[1] Dept. of Information Technology, Alexander TEI of Thessaloniki, 57400 Sindos, Greece
stoug@uom.gr, kdiamant@it.teithe.gr
[2] Dept. of Applied Informatics, University of Macedonia, 54006 Thessaloniki, Greece
gevan@uom.gr

Abstract. Although Support Vector Machines (SVMs) are considered effective supervised learning methods, their training procedure is time-consuming and has high memory requirements. Therefore, SVMs are inappropriate for large datasets. Many Data Reduction Techniques have been proposed in the context of dealing with the drawbacks of k-Nearest Neighbor classification. This paper adopts the concept of data reduction in order to cope with the high computational cost and memory requirements in the training process of SVMs. Experimental results illustrate that Data Reduction Techniques can effectively improve the performance of SVMs when applied as a preprocessing step on the training data.

Keywords: Support Vector Machines, k-NN classification, Data Reduction, Prototype Abstraction, Prototype Generation , Condensing

1 Introduction

The effectiveness, efficiency and scalability of machine learning and data mining algorithms are crucial research issues that have attracted the attention of both the industry and academia. Many proposed algorithms cannot handle high volumes of data that nowadays is easily available from several data sources. For those algorithms, data reduction[3] is an important preprocessing step.

In classification tasks data reduction processes are guided by the class labels. Many Data Reduction Techniques (DRTs) have been proposed in the context of dealing with the drawbacks of k-Nearest Neighbors (k-NN) classifier [9]. These drawbacks are: (i) the high computational cost during classification, (ii) the high memory requirements and (iii) the noise sensitivity of the classifier. However, that kind of data reduction has not been adopted by other classification methods which cannot manage large datasets.

[3] Data Reduction has two points of view: (i) item reduction, and, (ii) dimensionality reduction. We consider them from the first point of view.

A DRT can be either a Prototype Selection algorithm (PS) [10] or a Prototype Generation algorithm (PG) [20]. PS algorithms select representative instances from the training set. PG algorithms generate representatives by summarizing similar training instances. These representatives are called Prototypes. PS algorithms can be either editing or condensing. Editing aims at improving accuracy by removing noise, outliers and mislabeled instances and by smoothing the decision boundaries between classes. PG and PS-condensing algorithms try to build a small condensing set that represents the initial training data. Using a condensing dataset instead of the original dataset has the avail of low cost while accuracy remains almost as high as that achieved by using the original data. Please note that some PG and PS-condensing algorithms are called hybrid because they integrate the concept of editing.

To the best of our knowledge, DRTs have been used in the context of k-NN classification. There is no work that explores the application of data reduction on large datasets in order to render the usage of SVMs applicable on them. This is the key observation behind the motivation of the present work. Another motive is to check whether PG algorithms we proposed in the past can aid the development of fast and accurate SVM based classifiers.

This paper also contributes an experimental study on several datasets where SVM based classifiers, which are trained by the original training data and the corresponding condensing sets built by state-of-the-art DRTs, are compared to each other and against the corresponding k-NN classifiers. The paper reviews in detail the algorithms that are used in the experimental study. Our study reveals that the usage of DRTs leads to fast and accurate SVM-based classifiers.

The rest of this paper is organized as follows. Section 2 briefly reviews SVMs and the k-NN classifier and Section 3 presents in detail the PG and PS-condensing algorithms that we use in our experimental setup. Section 4 presents the experimental study and results. Finally, Section 5 concludes the paper.

2 Support Vector Machines and the k-NN classifier

2.1 Support Vector Machines

SVMs are supervised learning models introduced in 1995 by Cortes and Vapnik [7] although the roots of the idea lie in the theory of statistical learning introduced by Vapnik almost two decades earlier [21]. They are suitable for pattern classification but can be easily extended to handle nonlinear regression problems in which case they are known as Support Vector Regressors (SVRs). The separating surface offered by an SVM classifier maximizes the *margin*, i.e., the distance of the closest patterns to it. This helps the generalization performance of the model and in fact it is related to the idea of Structural Risk Minimization [23, 22] which avoids over-fitting. With the use of nonlinear kernel functions such as Gaussian (RBF) or n-th order polynomials, SVM models can produce nonlinear separating surfaces achieving very good performance in complex problems.

Due to their good generalization performance these models have become very popular with a wide range of applications, including document classification,

image classification, bio-informatics, handwritten character recognition, etc. One of the major drawbacks of these models is the memory and the computational complexity requirements for large datasets. The reason is that the separating surface is obtained by solving a quadratic programming problem involving an $N \times N$ matrix, where N is the number of items in the dataset. Although there are techniques that can reduce the complexity to $O(N^2)$ [5], the problem remains hard and the size of the problem can easily become prohibitively large calling for methods for data reduction such as the ones discussed in the following sections.

2.2 *k*-Nearest Neighbor classifier

The k-NN classifier [9] is an extensively used lazy (or instance-based) classification algorithm. Contrary to eager classifiers, it does not build any classification model. Some of its major properties are: (i) it is a quite simple and easy to implement algorithm, (ii) contrary to many other classifiers, it is easy to understand how a prediction has been made, (iii) it is analytically tractable and (iv) for $k = 1$ and unlimited instances the error rate is asymptotically never worse than twice the minimum possible, which is the Bayes rate [8].

The algorithm classifies a new instance by retrieving from the training set the k nearest instances to it. These instances are called neighbors. Subsequently, the algorithm assigns the new instance to the most common class among the classes of the k nearest neighbors. This class is called the major class. The process that indicates the major class is usually called nearest neighbors voting. Although any distance metric can be used, the Euclidean distance is the commonly-used distance metric. The k-NN classifier does not spend time in training any model. However, the classification step is time-consuming because in the worst case the algorithm must compute all distances between the new instance and all the training instances.

The selection of the value of k affects the accuracy of the classifier. The value of k that has the highest accuracy depends on the data. Its determination implies tuning via trial-and-error. Usually, large k values are appropriate for datasets with noise since they examine larger neighborhoods, whereas, small k values render the classifier noise-sensitive. In binary problems, an odd value for k should be used. Hence, possible ties in the nearest neighbors voting are avoided. In problems with more than two classes, ties are resolved by choosing a random "most common" class or the class voted by the nearest neighbor. The later is adopted in the experimental study of this paper.

3 Prototype Generation and Condensing algorithms

Several PG and PS-condensing algorithms are available in the literature. Here we review only the ones used in our experimental study. For the interested reader, abstraction and selection algorithms are reviewed, categorized and compared to each other in [20] and [10]. Other interesting reviews are presented in [19, 24, 4, 13].

3.1 Condensing Nearest Neighbor rule

The Condensing Nearest Neighbor (CNN) rule [11] is the earliest condensing algorithm. Its condensing set is built by the following simple idea. Instances that are far from decision boundaries ("internal") data area of a class can be removed without loss of accuracy. Thus, CNN-rule tries to keep only the instances that lie in the close-border areas. The close-border instances are selected as follows. Initially, an instance of the training set (TS) is moved to the condensing set (CS). CNN-rule uses the 1-NN rule and classifies the instances of TS by examining the instances of CS. If an instance is wrongly classified, it is probably close to decision boundaries. Therefore, it is moved from TS to CS. This procedure is repeated and if there are no moves from TS to CS in a complete pass of TS, the algorithm terminates.

CNN-rule is misled by noise. It wrongly selects "noisy" instances with their neighborhood. Consequently, noise affects the reduction rates. CNN-rule determines the number of the prototypes automatically, without user-defined parameters. Another property is that the multiple passes over the data guarantees that the removed training instances are correctly classified by 1-NN classifier in the context of the condensing set. A disadvantage is that CNN-rule builds a different condensing set by examining the same training instances in a different order.

3.2 The IB2 algorithm

IB2 is an one pass version of CNN-rule. IB2 is one of the Instance-Based Learning (IBL) algorithms presented in [2, 1]. Each training instance $x \in TS$ is classified by the 1-NN rule on the current CS. If x is classified correctly, x is discarded. Else, x is moved to CS.

IB2 determines the size of the condensing set automatically. However, the condensing set highly depends on the order of training instances. Since it is a one-pass algorithm, it is very fast. Also, IB2 does not guarantee that the removed instances can be correctly classified by the condensing set. In addition, it builds its condensing set in an incremental manner. This means than new training instances can update an existing condensing set without considering the "old" instances that had been used for the creation of the condensing set. Hence, IB2 can be applied in streaming environments where new instances are gradually available.

3.3 The AIB2 algorithm

In [15] we presented a PG variation of IB2. It is called Abstraction IB2 (AIB2) and inherits all the properties of IB2. AIB2 considers that the prototypes should be close to the center of the data area they represent. Contrary to IB2, AIB2 does not ignore the instances that were correctly classified. These instances contribute to the condensing set by repositioning the nearest prototype. To achieve this, each prototype has a weight value that denotes the number of instances it represents.

In an early step, a random training instance is placed in the condensing set and its weight becomes one. For each training instance x, AIB2 fetches its nearest prototype P from the current condensing set. If x has a class label different than the one of P, it is moved to the condensing set and plays the role of a prototype. Its weight becomes one. if x has the class label of P, the attributes of P are updated by taking into account the attributes of x and its weight. More formally, each attribute $attr(i)$ of P becomes $P_{attr(i)} \leftarrow \frac{P_{attr(i)} \times P_{weight} + x_{attr(i)}}{NN_{weight}+1}$. Thus, P moves towards x. Of course, the weight of P is increased by one and x is discarded.

3.4 The Reduction by Space Partitioning algorithms

Chen's algorithm The ancestor of the Reduction by Space Partitioning (RSP) algorithms is the PG algorithm proposed by Chen and Jozwik (Chen's algorithm) [6]. Chen's algorithm retrieves the instances that define the diameter of the training data, in other words the two most distant instances, a and b. Then, the algorithm splits the training data into two subsets. All the instances that are closer to a are moved to C_a. All other instances are placed in C_b. Subsequently, Chen's algorithm selects to split the non-homogeneous subset with the largest diameter. Non-homogeneous are called the subsets that have instances of more than one class. If there is no non-homogeneous subsets, the algorithm proceeds by spitting the homogeneous subsets. When the number of subsets is equal to a value specified by the user, the aforementioned procedure ends. The final step is the generation of prototypes. Each subset C is replaced by its mean instance. The class label of the mean instance is the major class in C. The mean instances constitute the condensing set.

The idea of splitting the homogeneous subset with the largest diameter is based on that this subset probably has more instances and thus, if it is split first, higher reduction will be achieved. Chen's algorithm generates the same condensing set regardless of the ordering of the instances. A drawback is that the user has to specify the number of subsets. Chen and Jozwik claim that this allows the user to define the trade-off between reduction rate and accuracy. However, the determination of this parameter implies costly trial-and-error procedures. Another weak point is that the instances that do not belong to the major class of the subset are not represented in the condensing set (they are ignored).

The RSP1 algorithm RSP1 [18] is similar to Chen's algorithm, but it does not ignore instances. It computes as many means as the number of distinct classes in the non-homogeneous subsets. RSP1 builds larger condensing sets than Chen's algorithm. However, it tries to improve the quality of the condensing set by taking into account all training instances.

The RSP2 algorithm RSP2 selects the subset that will be split first by examining the overlapping degree. The overlapping degree of a subset is the ratio

of the average distance between instances belonging to different classes and the average distance between instances that belong to the same class. This splitting criterion assumes that instances that belong to a class are as close to each other as possible whereas instances that belong to different classes lie as far as possible. As stated in [18], it is better to split the subset with the highest overlapping degree than that with the largest diameter.

The RSP3 algorithm RSP3 [18] is the only RSP algorithm (Chen's algorithm included) that builds its condensing set without any user specified parameter. RSP3 eliminates both weaknesses of Chen's algorithm. It splits all the non-homogeneous subsets. In other words, it terminates when all subsets become homogeneous. RSP3 can use either the diameter or the overlapping degree as spiting criterion. In effect, the selection of splitting criterion is an issue of secondary importance because all non-homogeneous subsets are eventually split. Certainly, the order of the training instances is irrelevant.

RSP3 generates many prototypes for close-border areas and few prototypes for "internal" areas. The size of the condensing set depends on the level of noise in the data. The higher the level of noise, the smaller subsets constructed and the lower reduction is achieved. Please note that the discovery of the most distant instances is a time-consuming procedure since all distances between the instances of the subset should be estimated. Thus, the usage of RSP3 may be prohibitive in the case of large datasets. Since we wanted to consider only non-parametric algorithms in our experimental study, we used only RSP3.

3.5 Reduction through Homogeneous Clusters

The RHC algorithm We have recently proposed the Reduction through Homogeneous Clusters (RHC) algorithm [16, 14]. It belongs to PG algorithms. Like RSP3, RHC is based on the concept of homogeneity but employs k-means clustering [12, 25]. Initially, the training data is considered as a non-homogeneous cluster in C. The algorithm computes a mean instance for each class in C. These mean instances are called class-means. Subsequently, RHC uses k-means clustering on C by adopting the class-means as initial means for k-means. The result is the creation of as many clusters as the number of discrete classes in C. This clustering process is applied on each non-homogeneous cluster. In the end, all clusters are homogeneous and each cluster contributes a prototype in the condensing set that is constructed by averaging the instances of the cluster.

RHC generates many prototypes for close-border areas and fewer for the "internal" areas. RHC uses the class-means as initial means for the k-means clustering in order to quickly find large homogeneous clusters. This property has the advantage of achieving a high reduction rate (the larger clusters discovered, the higher reduction rates achieved). Obviously, the instances that are noise can affect the reduction rates. Since RHC is based on k-means clustering, it is fast. Also, its condensing set does not depend on the ordering of the training data. The experimental study presented in [16, 14] shows that RHC has higher reduction

rates and is faster than RSP3 and CNN-rule, whereas accuracy remains high. Please note, that dRHC [16] is a variation of RHC that handles large datasets that cannot reside in the main memory.

The ERHC algorithm The Editing and Reduction through Homogeneous Clusters (ERHC) [17] algorithm is a simple variation of RHC that tries to deal with noisy data. ERHC differs from RHC on the following point: Whenever a homogeneous cluster with only one instance is discovered, ERHC discards it. Thus, the final condensing set contains the means of the homogeneous clusters that have more than one instance. Obviously, ERHC integrates an editing mechanism. It simultaneously removes noise and reduces the size of the training set. Therefore, it can be characterized as hybrid PG algorithm. The experimental study in [17] proves that this simple editing mechanism can improve classification performance when data contains noise.

4 Performance evaluation

4.1 Experimental setup

We conducted several experiments on thirteen datasets distributed by the KEEL repository[4] [3]. Their profiles are presented in Table 1. Five datasets do not contain noise. All the other datasets have noise of various levels (see column "Noise" in Table 1). We do not use any editing algorithm for noise removal. For each dataset, we built six condensing sets. They were built by applying the algorithms presented in Section 3. More specifically, we used CNN-rule, IB2, RSP3, RHC, ERHC and AIB2.

We trained several SVMs on the original training set (without data reduction) and for each condensing set by using several parameter values. Finally, we kept the most accurate SVMs. In Subsection 4.2, we report only the accuracy measurements for that SVM. The RBF kernel was used and the hyper-parameters γ, C where obtained through grid-search. Due to space restrictions, the parameter values we adopted are not reported[5].

For the five "noise-free" datasets, the k-NN classifier was run over the original training data and over the six condensing sets by setting $k = 1$. Most of the time $k = 1$ is the best choice for noise-free data. For the other eight datasets, we adopted four k values, namely, 1, 5, 9 and 13.

All measurements presented in Subsection 4.2 are average values obtained via a five-fold cross-validation. We used the Euclidean distance as distance metric. Since CNN-rule, IB2 and AIB2 depend on the order of class labels in the training set, we randomized all the datasets. Excluding CAR, we did not perform any other transformations. The CAR dataset has ordinal attributes. We transformed the attribute values into numerical values. Furthermore, we normalized to the interval [0-1] all attribute values of CAR.

[4] `http://sci2s.ugr.es/keel/datasets.php`
[5] They can be found in `http://users.uom.gr/~stoug/aiai2016.pdf`

Table 1. Dataset details

Dataset	Size (items)	Attributes	Classes	Noise
Letter Image Recognition (LIR)	20000	16	26	False
Pen-Digits (PD)	10992	16	10	False
Landsat Satellite (LS)	6435	36	6	True
Banana (BN)	5300	2	2	True
Balance (BL)	625	4	3	True
Texture (TXR)	5500	40	11	False
Yeast (YS)	1484	8	10	True
Phoneme (PH)	5404	5	2	False
MONK2 (MN2)	432	6	2	True
Twonorm (TN)	7400	20	2	True
Magic Gamma Telescope (MGT)	19020	10	2	True
Shuttle (SH)	58000	9	7	False
Car (CAR)	1728	6	4	True

4.2 Experimental results

We compared the six DRTs to each other by estimating the Preprocessing Cost (PC) and the Reduction Rate (RR) that they achieved. Since the larger the training set used, the higher the cost for k-NN classifier to classify a new item and the higher the cost of the training procedure of SVMs (it is at least $O(N^2)$ see Subsection 2.1), the RR measurements reflect the computational cost (the higher the RR, the lower the computational cost of k-NN classification and SVM training). Therefore, we do not include time measurements in our study.

Table 2 presents the RR and PC measurements. Best measurements are in bold. The last row shows the averages values. We observe that ERHC achieved the highest RR. This means that the SVM that uses the condensing set built by ERHC requites the least time for its training. AIB2 is the fastest DRT. It builds its condensing set by computing the fewest distances. On the other hand, RSP3 needs the highest computational cost in order to build its condensing set. In addition RSP3 seems to build the largest condensing sets. As expected, ERHC achieves higher RR than RHC and AIB2 is better in terms of RR and PC than IB2.

Tables 3 and 4 show the accuracy measurements achieved by the SVM and k-NN classifiers (Table 4 is the continuation of Table 3). Both tables contain seven rows for each dataset. Each row represents the different versions of the same dataset. The first one concerns the original data (i.e., without data reduction). The other six rows concern the condensing set constructed by the DRTs. Each column of the table concerns a classifier. In particular, the third column concerns the SVM classifiers while the other columns concern the k-NN classifiers. The best accuracy measurements of the different classifiers are in bold. The best accuracy among the different condensing sets is emphasized with italic style.

The results depicted by both tables are quite interesting. Almost in all cases, SVM classifiers are more accurate than the k-NN classifier. In addition, all DRTs

Table 2. Comparison of DRT algorithms in terms of Reduction Rate (RR(%)) and Preprocessing Cost (PC (millions of distance computations))

Dataset		CNN	IB2	RSP3	RHC	ERHC	AIB2
LIR	RR	83.54	85.66	61.98	88.08	**92.03**	88.12
	PC	163.03	23.37	326.52	41.85	41.85	**20.10**
PD	RR	95.36	96.23	89.22	96.52	**97.45**	97.19
	PC	11.75	1.78	86.66	2.88	2.88	**1.38**
LS	RR	80.22	84.62	73.19	89.84	**92.95**	86.72
	PC	17.99	2.22	37.70	**1.69**	**1.69**	1.92
BN	RR	77.44	83.27	75.21	79.68	**90.33**	83.40
	PC	11.49	1.58	18.76	**0.56**	**0.56**	1.53
BL	RR	65.72	69.36	64.64	78.00	**86.68**	70.36
	PC	0.21	**0.04**	0.3	0.05	0.05	**0.04**
TXR	RR	91.90	93.33	83.31	94.71	**95.94**	94.95
	PC	5.65	0.84	27.63	3.63	3.63	**0.66**
YS	RR	32.68	44.82	27.36	49.83	**79.34**	46.94
	PC	1.41	0.39	2.12	0.84	0.84	**0.37**
PH	RR	76.04	80.85	69.94	80.71	**88.05**	81.75
	PC	13.45	1.96	20.31	**0.66**	**0.66**	1.84
MN2	RR	87.23	91.68	61.33	96.47	**96.76**	92.54
	PC	0.04	0.006	0.13	0.007	0.007	**0.005**
TN	RR	82.09	88.25	84.56	96.63	**97.58**	93.44
	PC	22.13	2.07	37.13	1.64	1.64	**1.10**
MGT	RR	60.08	70.60	53.70	73.76	**84.46**	71.90
	PC	281.49	34.61	511.67	**4.08**	**4.08**	33.05
SH	RR	99.37	99.44	98.59	99.55	**99.69**	99.46
	PC	45.30	8.26	1,7410.18	16.83	16.83	**7.89**
CAR	RR	75.82	81.61	68.65	85.87	**90.31**	83.63
	PC	1.50	0.19	1.94	0.18	0.18	**0.17**
AVG	RR	77.50	82.29	70.13	85.36	**91.66**	83.88
	PC	44.26	5.95	89.24	5.76	5.76	**5.36**

seem to not affect accuracy achieved by SVMs. In most cases, a SVM trained by any condensing set is as accurate as the SVM trained by the initial training set. In eight datasets, the SVMs trained by the condensing set of RSP3 are the most accurate classifier. However, RSP3 has the highest PC and the lowest RR measurements. In the cases of the rest five datasets, the most accurate classifier is the SVM built by the condensing set of CNN-rule. The accuracy achieved by IB2 is close enough to that of CNN-rule, but IB2 is faster and achieved higher RR. A final comment is that the PG and PS-condensing algorithms can effectively be used for speeding-up the training process of SVMs without sacrifying accuracy. Furthermore, we observe that, in the case of SVMs, the editing mechanism of ERHC is not as effective as it is when the k-NN classifier is used. In addition, although AIB2 achieves higher accuracy than IB2 in the case of k-NN classification, it is not true in the case of SVMs. Consequently, for SVMs, ERHC and AIB2 are not efficient extensions of RHC and IB2 respectively.

5 Conclusions

This paper demonstrated that the DRTs proposed for the k-NN classifier can also be applied for speeding-up SVMs. More specifically, the experimental measurements of our study showed that the usage of a DRT can reduce the time needed for the training process of SVMs without negatively affecting accuracy.

Table 3. Comparison of DRT algorithms in terms of accuracy (%) - Datasets LIR through MN2

Dataset	DRT	SVM	1-NN	5-NN	9-NN	13-NN
LIR	None	**97.58**	95.83	-	-	-
	CNN	**95.10**	92.84	-	-	-
	IB2	**94.75**	91.98	-	-	-
	RSP3	***96.51***	95.43	-	-	-
	RHC	**93.80**	93.59	-	-	-
	ERHC	92.60	**92.69**	-	-	-
	AIB2	93.71	**94.12**	-	-	-
PD	None	**99.65**	99.35	-	-	-
	CNN	**99.21**	98.68	-	-	-
	IB2	**99.02**	98.04	-	-	-
	RSP3	***99.50***	99.05	-	-	-
	RHC	**98.81**	98.30	-	-	-
	ERHC	**98.84**	98.63	-	-	-
	AIB2	**98.74**	98.33	-	-	-
LS	None	**92.40**	90.60	90.69	90.62	90.21
	CNN	**90.83**	88.21	89.99	89.39	88.00
	IB2	**88.73**	86.87	88.45	87.55	86.23
	RSP3	***91.14***	90.57	90.16	89.64	89.50
	RHC	88.95	88.95	**89.54**	88.21	86.05
	ERHC	88.37	**89.01**	88.90	86.81	84.26
	AIB2	88.19	**89.40**	87.69	86.03	84.34
BN	None	**90.57**	86.91	89.02	89.85	89.87
	CNN	***90.53***	85.62	88.15	89.09	88.77
	IB2	**90.06**	83.81	87.57	88.08	88.00
	RSP3	**90.30**	84.00	87.83	89.11	88.91
	RHC	**90.25**	83.28	87.23	88.19	88.38
	ERHC	**90.23**	88.00	89.09	88.64	88.00
	AIB2	**90.49**	82.96	87.89	88.57	89.26
BL	None	**90.96**	78.40	84.16	87.84	88.32
	CNN	***95.36***	70.88	76.32	82.72	84.16
	IB2	**95.36**	70.72	77.28	81.60	83.20
	RSP3	**94.72**	73.28	76.64	82.88	82.88
	RHC	**93.44**	68.04	75.36	82.56	84.32
	ERHC	**89.60**	76.00	83.52	83.68	83.68
	AIB2	**94.72**	68.64	77.12	83.20	81.92
TXR	None	**99.84**	99.02	-	-	-
	CNN	**99.58**	97.16	-	-	-
	IB2	**99.56**	96.35	-	-	-
	RSP3	***99.69***	98.29	-	-	-
	RHC	**99.24**	97.04	-	-	-
	ERHC	**99.10**	97.36	-	-	-
	AIB2	**99.45**	97.69	-	-	-
YS	None	**60.11**	52.02	57.01	58.15	59.43
	CNN	**59.84**	49.06	52.97	55.66	56.94
	IB2	**59.03**	46.02	52.29	55.53	55.66
	RSP3	***59.84***	50.47	54.99	57.08	57.75
	RHC	**58.89**	48.85	52.29	54.65	54.92
	ERHC	**59.09**	53.17	56.00	55.86	56.80
	AIB2	**58.22**	48.25	51.48	56.06	56.81
PH	None	89.19	**90.10**	-	-	-
	CNN	87.56	***87.82***	-	-	-
	IB2	**86.47**	85.57	-	-	-
	RSP3	**87.10**	86.94	-	-	-
	RHC	**85.88**	85.59	-	-	-
	ERHC	86.08	**86.57**	-	-	-
	AIB2	**85.83**	84.92	-	-	-
MN2	None	**100.00**	90.51	99.31	98.84	99.07
	CNN	**97.21**	95.84	90.97	84.03	84.26
	IB2	**94.66**	93.75	81.47	80.33	78.00
	RSP3	***99.08***	91.22	98.38	97.69	96.76
	RHC	91.43	**94.68**	80.09	67.12	47.12
	ERHC	90.26	**95.14**	77.53	63.65	47.12
	AIB2	**93.04**	91.43	85.65	78.23	66.70

Table 4. Comparisson of DRT algorithms in terms of Accuracy (%) - Datasets TN through CAR

Dataset	DRT	SVM	1-NN	5-NN	9-NN	13-NN
TN	None	**97.89**	94.88	96.91	97.31	97.38
	CNN	***97.84***	92.00	95.47	96.50	96.82
	IB2	**97.81**	89.15	94.95	95.87	96.45
	RSP3	**97.70**	92.68	96.30	96.88	97.31
	RHC	**97.39**	88.69	95.74	96.69	97.10
	ERHC	**97.45**	91.53	96.50	96.92	97.07
	AIB2	**97.73**	93.47	96.69	97.28	97.28
MGT	None	**83.79**	78.14	80.48	80.84	81.12
	CNN	***83.90***	74.54	78.63	79.65	80.24
	IB2	**83.38**	71.97	76.84	78.50	79.11
	RSP3	**83.71**	74.96	78.90	80.15	80.52
	RHC	**83.08**	71.97	76.67	77.83	78.94
	ERHC	**83.12**	77.01	79.64	79.86	79.86
	AIB2	**83.13**	73.36	77.40	78.36	78.89
SH	None	**99.84**	99.82	-	-	-
	CNN	99.66	**99.76**	-	-	-
	IB2	99.62	**99.73**	-	-	-
	RSP3	***99.81***	99.75	-	-	-
	RHC	**99.64**	98.01	-	-	-
	ERHC	**99.64**	98.04	-	-	-
	AIB2	99.74	**99.72**	-	-	-
CAR	None	**94.10**	85.53	89.75	90.39	89.76
	CNN	**93.28**	84.95	88.14	84.72	81.48
	IB2	**92.24**	84.43	86.11	84.20	82.41
	RSP3	***93.75***	85.59	89.70	89.12	88.14
	RHC	**87.10**	82.75	77.66	75.29	71.81
	ERHC	**86.9**8	82.69	79.92	78.00	74.76
	AIB2	**91.72**	87.09	87.56	85.30	82.40

Although the particular DRTs have been proposed for speeding up the k-NN classifier, our study illustrated that the benefits are larger when SVMs are used. The experimental results showed that in contrast to the k-NN classifier that can be affected by data reduction, the accuracy of SVMs is not affected.

References

1. Aha, D.W.: Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. Int. J. Man-Mach. Stud. 36(2), 267–287 (Feb 1992), http://dx.doi.org/10.1016/0020-7373(92)90018-G
2. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. 6(1), 37–66 (Jan 1991), http://dx.doi.org/10.1023/A:1022689900470
3. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Multiple-Valued Logic and Soft Computing 17(2-3), 255–287 (2011)
4. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. Data Min. Knowl. Discov. 6(2), 153–172 (Apr 2002), http://dx.doi.org/10.1023/A:1014043630878
5. Chapelle, O.: Training a support vector machine in the primal. Neural Computation 19, 1155–1178 (2007)

6. Chen, C.H., Jóźwik, A.: A sample set condensation algorithm for the class sensitive artificial neural network. Pattern Recogn. Lett. 17(8), 819–823 (Jul 1996), http://dx.doi.org/10.1016/0167-8655(96)00041-4
7. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning 20(3), 273–297 (1995)
8. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theor. 13(1), 21–27 (Sep 2006), http://dx.doi.org/10.1109/TIT.1967.1053964
9. Dasarathy, B.V.: Nearest neighbor (NN) norms : NN pattern classification techniques. IEEE Computer Society Press (1991)
10. 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 (Mar 2012), http://dx.doi.org/10.1109/TPAMI.2011.142
11. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory 14(3), 515–516 (1968)
12. McQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proc. of 5th Berkeley Symp. on Math. Statistics and Probability. pp. 281– 298. Berkeley, CA : University of California Press (1967)
13. Olvera-López, J.A., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., Kittler, J.: A review of instance selection methods. Artif. Intell. Rev. 34(2), 133–143 (Aug 2010), http://dx.doi.org/10.1007/s10462-010-9165-y
14. Ougiaroglou, S., Evangelidis, G.: Efficient dataset size reduction by finding homogeneous clusters. In: Proceedings of the Fifth Balkan Conference in Informatics. pp. 168–173. BCI '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2371316.2371349
15. Ougiaroglou, S., Evangelidis, G.: Efficient data abstraction using weighted IB2 prototypes. Comput. Sci. Inf. Syst. 11(2), 665–678 (2014), http://dx.doi.org/10.2298/CSIS1402120360
16. Ougiaroglou, S., Evangelidis, G.: Rhc: a non-parametric cluster-based data reduction for efficient k-nn classification. Pattern Analysis and Applications 19(1), 93–109 (2014), http://dx.doi.org/10.1007/s10044-014-0393-7
17. Ougiaroglou, S., Evangelidis, G.: Efficient editing and data abstraction by finding homogeneous clusters. Annals of Mathematics and Artificial Intelligence 76(3), 327–349 (2015), http://dx.doi.org/10.1007/s10472-015-9472-8
18. Sánchez, J.S.: High training set size reduction by space partitioning and prototype abstraction. Pattern Recognition 37(7), 1561–1564 (2004)
19. Toussaint, G.: Proximity graphs for nearest neighbor decision rules: Recent progress. In: 34th Symposium on the INTERFACE. pp. 17–20 (2002)
20. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. Trans. Sys. Man Cyber Part C 42(1), 86–100 (Jan 2012), http://dx.doi.org/10.1109/TSMCC.2010.2103939
21. Vapnik, V.: Estimation of dependencies based on empirical data. Nauka, Moscow (1979), english translation: Springer Verlag, New York, 1982
22. Vapnik, V.: Statistical learning theory. Wiley New York (1998)
23. Vapnik, V., Chervonenkis, A.: Theory of pattern recognition (1974)
24. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Mach. Learn. 38(3), 257–286 (Mar 2000), http://dx.doi.org/10.1023/A:1007626913721
25. Wu, J.: Advances in K-means Clustering: A Data Mining Thinking. Springer Publishing Company, Incorporated (2012)