Using the geometrical distribution of prototypes for training set condensing ⋆

M. Lozano, J.S. Sánchez, and F. Pla

Dept. Lenguajes y Sistemas Informáticos, Universitat Jaume I
Campus Riu Sec, 12071 Castellón, Spain
[lozano,sanchez,pla]@uji.es

Abstract. In this paper, some new approaches to training set size reduction are presented. These schemes basically consist of defining a small number of prototypes that represent all the original instances. Although the ultimate aim of the algorithms proposed here is to obtain a strongly reduced training set, the performance is empirically evaluated over nine real datasets by comparing the reduction rate and the classification accuracy with those of other condensing techniques.

1 Introduction

Currently, in many domains (e.g., in multispectral images, text categorisation, and retrieval of multimedia databases) the size of the datasets is so extremely large that real-time systems cannot afford the time and storage requirements to process them. Under these conditions, classifying, understanding or compressing the available information can become a very problematic task. This problem is specially dramatic in the case of using some distance-based learning algorithm, such as the Nearest Neighbour (NN) rule [7]. The basic NN scheme must search through all the available training instances (large memory requirements) to classify a new input sample (slow during classification). On the other hand, since the NN rule stores every prototype in the training set (TS), noisy instances are stored as well, which can considerably degrade the classification accuracy.

Among the many proposals to tackle this problem, a traditional method consists of removing some of the training prototypes. In the Pattern Recognition literature, those methods leading to reduce the TS size are generally referred to as *prototype selection* [9]. Two different families of prototype selection methods can be defined. First, the *editing* approaches eliminate erroneously labelled prototypes from the original TS and "clean" possible overlapping among regions from different classes. Second, the *condensing* algorithms aim at selecting a small subset of prototypes without a significant degradation of classification accuracy.

Wilson introduced the first editing method [15]. Briefly, it consists of using the k-NN rule to estimate the class of each prototype in the TS, and removing

⋆ This work has been supported by grants TIC2003-08496-C04-03 and CPI2001-2956-C02-02 from CICYT MCT and projects IST-2001-37306 from the European Union and P1-1B2002-07 from *Fundació Caixa Castelló-Bancaixa*.

those whose class label does not agree with that of the majority of its k-NN. This algorithm tries to eliminate mislabelled prototypes from the TS as well as those close to the decision boundaries. Subsequently, many researchers have addressed the problem of editing by proposing alternative schemes [1, 7, 9, 16].

Within the condensing perspective, the many existing proposals can be categorised into two main groups. First, those schemes that merely select a subset of the original prototypes [1, 8, 10, 13, 14] and second, those that modify them [2, 3, 6]. One problem related with using the original instances is that there may not be any vector located at the precise point that would make the most accurate learning algorithm. Thus, prototypes can be artificially generated to exist exactly where they are needed.

This paper focuses on the problem of appropriately reducing the TS size by selecting a subset of prototypes. The primary aim of the proposal presented in this paper is to obtain a considerable size reduction rate, but without an important decrease in classification accuracy.

The structure of the rest of this paper is as follows. Section 2 briefly reviews a set of TS size reduction techniques. The condensing algorithms proposed here are introduced in Section 3. The databases used and the experiments carried out are described in Section 4. Results are shown and discussed in Section 5. Finally, the main conclusions along with further extensions are depicted in Section 6.

2 Prototype Selection

The problem of prototype selection is primarily related to prototype deletion as irrelevant and harmful prototypes are removed. This is the case, e.g., of Hart's condensing [10], Tomek's condensing [13], proximity graph-based condensing [14] and MCS scheme of Dasarathy [8], in which only critical prototypes are retained. Some other algorithms artificially generate prototypes in locations accurately determined in order to reduce the TS size. Within this category, we can find the algorithms presented by Chang [3] and by Chen and Józwik [6].

Hart's algorithm [10] is based on reducing the set size by eliminating prototypes. It is the earliest attempt at minimising the size set by retaining only a consistent subset. A consistent subset, S, of a TS, T, is a subset that correctly classifies every prototype in T using the 1-NN rule. The minimal consistent subset is the most interesting to minimise the cost of storage and the computing time. Hart's condensing does not guarantee finding the minimal subset.

Tomek's condensing [13] consists of Hart's condensing, adding an appropriate selection strategy. It consists of selecting a subset with the boundary prototypes (the closest to the decision boundaries). Some negative aspects are its computational cost $\mathrm{O}(\mathrm{N}^3)$ and that the boundary subset chosen is not consistent.

The Voronoi's condensing [14] is the only scheme able to obtain a reduced set that satisfies the consistency criteria with respect to a) the decision boundaries, and b) the TS. Despite this, the Voronoi condensing presents two important problems. First its high computational cost, since the calculation of the Voronoi

Diagram associated to the prototypes set is required. And second, it deals with every representation space region in the same way.

In [14] an alternative similar to Voronoi condensing is proposed, with the aim of solving these two important drawbacks. This new alternative is based on two proximity graph models: the Gabriel Graph and the Relative Neighbourhood Graph. The main advantages of this algorithm are that a) it ignores the TS prototypes that maintain the decision boundaries out of the interest region, and b) it reduces the computational cost ($O(dN^2)$). Both condensed sets are not consistent with respect to the decision boundaries and therefore, they are not consistent with respect to the TS either.

Aha et al. [1] presented the incremental learning schemes *IB1-IB4*. In particular, *IB3* addresses the problem of keeping noisy prototypes by retaining only acceptable misclassified cases.

Within the group of condensing proposals that are based on generating new prototypes, Chang's algorithm [3] consists of repeatedly attempting to merge the nearest two existing prototypes into a new single one. Two prototypes p and q are merged only if they are from the same class and, after replacing them with prototype z, the consistency property can be guaranteed.

Chen and Józwik [6] proposed an algorithm which consists of dividing the TS into some subsets using the concept of *diameter of a set* (distance between the two farthest points). It starts by partitioning the TS into two subsets by the middle point between the two farthest cases. The next division is performed for the subset that contains prototypes from different classes. If more than one subset satisfies this condition, then that with the largest diameter is divided. The number of partitions will be equal to the number of instances initially defined. Finally, each resulting subset is replaced by its centroide, which will assume the same class label as the majority of instances in the corresponding subset.

Recently, Ainslie and Sánchez introduced the family of *IRSP* [2], which are based on the idea of Chen's algorithm. The main difference is that by Chen's any subset containing prototypes from different classes could be chosen to be divided. On the contrary, by *IRSP4*, the subset with the highest overlapping degree (ratio of the average distance between prototypes from different classes, and the average distance between instances from the same class) is split. Furthermore, with *IRSP4* the splitting process continues until every subset is homogeneous.

3 New Condensing Algorithms

The geometrical distribution among prototypes in a TS can become even more important than just the distance between them. In this sense, the *surrounding neighbourhood-based rules* [12] try to obtain more suitable information about prototypes in the TS and specially, for those being close to decision boundaries. This can be achieved by taking into account not only the proximity of prototypes to a given input sample but also their *symmetrical distribution* around it.

Chaudhuri [5] proposed a neighbourhood concept, the Nearest Centroide Neighbourhood (NCN), a particular realization of the surrounding neighbourhood. Let p be a given point whose k NCN should be found in a TS, $X=\{x_1, .., x_n\}$. These k neighbours can be searched for by an iterative procedure like the next:

1. The first NCN of p corresponds to its NN, q_1.
2. The i-th NCN, q_i, $i \geq 2$, is such that the centroide of this and previously selected NCN, $q_1, .., q_i$ is the closest to p.

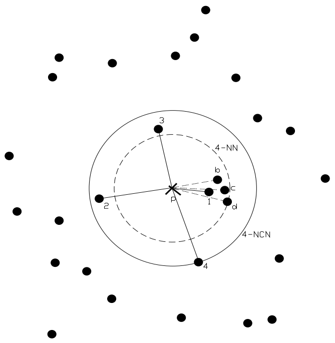

Fig. 1. Example of the NCN concept.

The neighbourhood obtained satisfies some interesting properties that can be used to reduce the TS size: the NCN search method is incremental and the prototypes around a given sample have a geometrical distribution that tends to surround the sample. It is also important to note that in general, the region of influence of the NCN results bigger than that of the NN, as can be seen in Fig. 1.

3.1 The Basic Algorithm

The TS size reduction technique here proposed rests upon the NCN algorithm. NCN search is used as an exploratory tool to bring out how prototypes in the data set are geometrically distributed. The use of the NCN of a given sample can provide local information about what is the shape of the probability class distribution depending on the nature and class of its NCN. The rationale behind it is that prototypes belonging to the same class are located in a neighbouring area and could be replaced by a single representative without significantly affecting the original boundaries. The main reason to use the NCN instead of the NN is to benefit from the aforementioned properties: that the NCN covers a bigger region, and that these neighbours are located in an area of influence around a given sample which is compensated in terms of their geometrical distribution.

The algorithm attempts to replace a group of neighbouring prototypes from the same class by a representative. In order to decide which group of prototypes is to be replaced, the NCN of each prototype p in the TS is computed until reaching a neighbour from a different class than that of p. The prototype with the largest number of neighbours is defined as the representative of its corresponding group, which lies in the area of influence defined by the NCN distribution and consequently, all its members can be now removed from the TS. Another possibility is to replace the group by its centroide. In this case, the reduction of the data set is done by introducing new samples. For each prototype remaining in the set, we update the number of its neighbours if some were previously eliminated as belonging to the group of an already chosen representative.

This is repeated until there is no group of prototypes to be replaced. The basic scheme has been named *MaxNCN*. A further extension consists of iterating the general process until no more prototypes are removed from the TS. The iterative version can be written as follows:

Algorithm 1 *IterativeMaxNCN*

```
while $eliminated\_prototypes > 0$ do
  for $i = eachprototype(TS)$ do
    $neighbours\_number[i] = 0$
    $neighbour = next\_neighbour(i)$
    while $neighbour.class == i.class$ do
      $neighbours\_vector[i] = Id(neighbour)$
      $neighbours\_number[i] = neighbours\_number[i] + 1$
      $neighbour = next\_neighbour(i)$
    end while
  end for
  while $Max\_neighbours() > 0$ do
    $EliminateNeighbours(id\_Max\_neighbours)$
  end while
end while
```

Apart from the basic *MaxNCN* and its iterative version, other alternatives have been implemented and tested: *IterativekNeighbours*, *Centroide* and *WeightedCentroide* among others. *IterativekNeighbours* is similar to Algorithm 1. The main difference relies on the number of neighbours allowed to be represented by a prototype: k. One of its main properties is that the limit of neighbours can be selected, depending on the TS size (here k is a percentage of the TS size).

In *Centroide*, the main difference to Algorithm 1 is that instead of using an original prototype as a representative, it computes the respective centroide of the NCN. The rationale behind this is that a new artificial prototype could represent better a neighbourhood because it can be placed in the best location.

WeightedCentroide uses the same idea, but each centroide is calculated weighting each prototype by the number of neighbours that it represents.

3.2 The Consistent Algorithm

Over the basic algorithm described in the previous subsection, we tried to do an important modification. The aim is to obtain a consistent condensed subset. The primary idea is that if the subset is consistent with the TS, a better classification should be obtained. Using the *MaxNCN* algorithm, some prototypes in the decision boundaries are removed because of the condensing order. We try to solve this problem by a new consistent approach. Other alternatives have been implemented. Two of them are presented here. The simplest one consists of applying *MaxNCN* and, after that, estimating the class of each prototype in the TS by NN, using the reduced set obtained. Every prototype misestimated is added to the reduced set. This algorithm has been here named *Consistent*.

Reconsistent is based on the same idea as *Consistent*. In this case, the new prototypes to be added will previously be condensed using as reference the original TS. Algorithmically, it can be written as it is shown in Algorithm 2.

Algorithm 2 *Reconsistent*

```
for $i = eachprototype(TS)$ do
  $neighbours\_number[i] = 0$
  $neighbour = next\_neighbour(i)$
  while $neighbour.class == i.class$ do
    $neighbours\_vector[i] = Id(neighbour)$
    $neighbours\_number[i] = neighbours\_number[i] + 1$
    $neighbour = next\_neighbour(i)$
  end while
end for
while $Max\_neighbours() > 0$ do
  $EliminateNeighbours(id\_Max\_neighbours)$
end while
$count = 0$
for $i = eachprototype(TS)$ do
  if $Classify(i)! = i.class$ then
    $count = count + 1$
    $incorrect\_class[count] = i$
  end if
end for
for $i = eachprototype(incorrect\_class[])$ do
  $neighbours\_number\_inc[i] = 0$
  $neighbour\_inc = next\_neighbour\_inc(i)$
  while $neighbour\_inc.class == i.class$ do
    $neighbours\_vector\_inc[i] = Id(neighbour\_inc)$
    $neighbours\_number\_inc[i] = neighbours\_number\_inc[i] + 1$
    $neighbour\_inc = next\_neighbour\_inc(i)$
  end while
end for
while $Max\_neighbours\_inc() > 0$ do
  $EliminateNeighbours\_inc(id\_Max\_neighbours\_inc)$
end while
$AddCondensedIncToCondensedTS()$
```

4 Description of Databases and Experiments

Nine real data sets (Table 1) have been taken from the UCI Repository [11] to assess the behaviour of the algorithms introduced in this paper. The experiments have been conducted to compare *MaxNCN*, *IterativeMaxNCN*, *IterativekNeighbours*, *Centroide*, *WeightedCentroide*, *Consistent* and *Reconsistent*, among other algorithms to Chen's scheme, *IRSP4* and Hart's condensing, in terms of both TS size reduction and classification accuracy (using 1-NN rule) for the condensed set.

The algorithms proposed in this paper, as in the case of Chen's, *IRSP4*, *MaxNCN* and *IterativeMaxNCN* need to be applied in practice to overlap-free (no overlapping among different class regions) data sets. Thus, as a general rule and according to previously published results [2,16], the Wilson's editing has been considered to properly remove possible overlapping between classes. The parameter involved (k) has been obtained in our experiments by performing a five-fold cross-validation experiment using only the TS and computing the average classification accuracies for different values of k and comparing them with the "no editing" option. The best edited set (including the non-edited TS) is thus selected as input for the different condensing schemes.

Data set	No. classes	No. features	TS size	Test set size
Cancer	2	9	546	137
Pima	2	6	615	153
Glass	6	9	174	40
Heart	2	13	216	54
Liver	2	6	276	69
Vehicle	4	18	678	168
Vowel	11	10	429	99
Wine	3	13	144	34
Phoneme	2	5	4324	1080

Table 1. Data sets used in the experiments.

5 Experimental Results and Discussion

Table 2 reports the 1-NN accuracy results obtained by using the best edited TS and the different condensed sets. Values in brackets correspond to the standard deviation. Analogously, the reduction rates with respect to the edited TS are provided in Table 3. The average values for each method are also included. Several comments can be made from the results in these tables. As expected, classification accuracy strongly depends on the condensed set size. Correspondingly, *IRSP4*, Hart's algorithm, *Consistent* and *Reconsistent* obtain the highest classification accuracy almost without exception for all the data sets, but they also retain more prototypes than Chen's scheme, *MaxNCN* and *IterativeMaxNCN*.

It is important to note that, in terms of reduction rate, *IterativeMaxNCN* is the best. Nevertheless, it also obtains the worst accuracy. On the contrary, *IRSP4* shows the highest accuracy but the lowest reduction rate. Thus, looking for a balance between accuracy and reduction, one can observe that the best options are Hart's, Chen's, the plain *MaxNCN* and the *Reconsistent* approach. In

	Edited	Chen	IRSP4	Hart	Iterat.	MaxNCN	Cons.	Recons.
Cancer	95.61	96.78	93,55	94,61	68,60	89,92	94,14	92,39
	(2.48)	(1.25)	(3,70)	(2,94)	(3,42)	(4,61)	(2,64)	(4,36)
Pima	67.32	73.64	72,01	73,31	53,26	67,71	73,05	71,74
	(4.64)	(2.85)	(4,52)	(3,69)	(5,80)	(5,45)	(3,62)	(5,93)
Glass	71.40	67.18	71,46	67,91	57,19	66,65	69,08	69,08
	(3.78)	(3.90)	(3,13)	(4,60)	(9,69)	(6,28)	(3,90)	(3,90)
Heart	58.17	61.93	63,01	62,87	58,16	59,92	63,96	63,59
	(5.93)	(5.22)	(5,11)	(4,27)	(7,26)	(5,53)	(6,87)	(5,98)
Liver	65.79	59.58	63,89	62,40	53,31	60,65	63,23	62,13
	(8.72)	(5.15)	(7,73)	(5,76)	(8,55)	(6,74)	(3,55)	(6,76)
Vehicle	64.41	58.56	63,47	62,17	55,20	59,33	62,76	62,65
	(2.11)	(2,46)	(1,96)	(2,16)	(4,42)	(2,17)	(1,04)	(1,88)
Vowel	97.90	60.16	96,02	90,74	78,63	90,73	94,93	94,73
	(1.23)	(9.27)	(1,77)	(2,30)	(5,18)	(1,78)	(1,63)	(1,53)
Wine	73.05	69.31	69,66	71,71	62,50	60,77	69,05	68,56
	(2.96)	(7.31)	(3,47)	(6,72)	(6,65)	(6,19)	(6,40)	(6,18)
Phoneme	70.26	70.03	71,60	71,04	65,06	70,00	72,17	70,96
	(7.52)	(9.14)	(8,74)	(7,90)	(7,57)	(8,05)	(7,72)	(7,13)
Average	73.77	68.57	73,85	72,97	61.32	69,52	73,60	72,87
	(4.38)	(5.17)	(4,46)	(4,48)	(9,95)	(5,20)	(4,15)	(4,85)

Table 2. Experimental results: 1-NN classification accuracy.

	Chen	IRSP4	Hart	Iterat.	MaxNCN	Cons.	Recons.
Cancer	98.79	93,72	93,09	99,11	96,10	86,91	94,09
Pima	90.61	70,03	79,04	95,99	85,35	73,01	80,19
Glass	67.58	32,71	51,33	73,13	62,15	47,43	50,00
Heart	85.18	55,80	67,22	92,53	78,35	64,18	69,59
Liver	82.97	45,41	63,20	91,21	74,83	57,85	65,65
Vehicle	65.79	35,60	45,98	74,85	56,59	40,28	44,71
Vowel	79.64	39,54	75,97	84,23	75,09	72,21	73,11
Wine	86.75	73,13	78,79	89,03	84,83	65,63	79,71
Phoneme	94.51	69,90	87,91	98,16	90,88	83,06	88,26
Average	83.54	57,32	71,39	88,69	78,24	65,62	71,70

Table 3. Experimental results: set size reduction rate.

particular, *MaxNCN* provides an average accuracy of 69,52% (only 4 points less than *IRSP4*, which is the best option in accuracy) with an average reduction of 78,24% (approximately 20 points higher than *IRSP4*). Results given by Chen's algorithm are similar to those of the *MaxNCN* procedure in accuracy, but 5 points higher in reduction percentage. The *Reconsistent* approach provides similar results to Hart's algorithm: an average accuracy of 72,87% (only 0,93 less than *IRSP4*) with an average reduction rate of 71,70% (around 14 points higher).

In order to assess the performance of these two competing goals simultaneously, Fig. 2 represents the normalised Euclidean distance between each pair (accuracy, reduction) and the ideal case (1, 1), in such a way that the "best" ap-

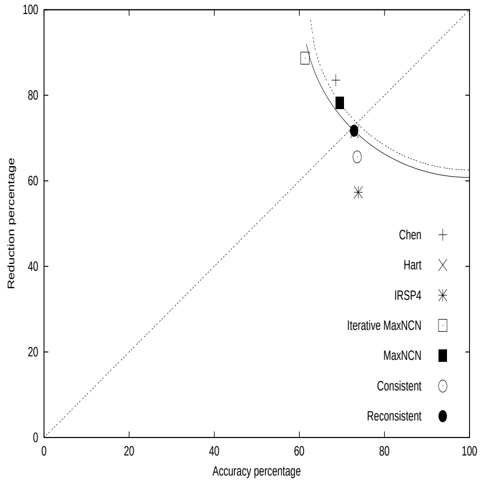

Fig. 2. Averaged accuracy and reduction rates.

proach can be deemed as the one that is nearer (1, 1). This technique is usually referred to as *Data Envelopment Analysis* [4] Among approaches with similar distances to (1, 1), the best ones are those nearer the diagonal, because they get a good balance between accuracy and reduction. Thus, it is possible to see that the proposed *Reconsistent*, along with *MaxNCN*, Hart's and Chen's algorithms represent a good trade-off between accuracy and reduction.

With respect to the other algorithms exposed here, as they are not considered so important as the ones compared until now, they are not drawn in Fig. 2, in order to obtain a more comprehensible representation. Anyway, their results are commented here. *IterativekNeighbours* obtains a good reduction rate but not the best accuracy rate (similar to *IterativeMaxNCN*). Anyway, comparing it to Hart, the positive difference in reduction is bigger than the negative difference in accuracy.

Centroide obtains more or less the same reduction as *IterativekNeighbours*, but its accuracy rate is a little bit higher. *WeightedCentroide* obtains more or less the same reduction rate as *IterativekNeighbours* and *Centroide*, and also a little bit higher accuracy rate than them.

Finally, it is to be noted that several alternatives to the algorithms here introduced have also been analysed, although some of them had a behaviour similar to that of *MaxNCN*. Other alternatives, as for example *MaxNN*, consisting of using the NN instead of the NCN, have a performance systematically worst.

Many algorithms have been tested. In Fig. 2 an imaginary curve formed by the results for some of them can be observed. It seems that when the classification accuracy increases, the reduction percentage decreases; and when the reduction percentage increases, the classification accuracy decreases. It makes sense because there should be some limit to the reduction. That is, a set can not be reduced as much as we want without influencing the classification accuracy.

6 Concluding Remarks

In this paper, some new approaches to TS size reduction have been introduced. These algorithms primarily consist of replacing a group of neighbouring prototypes that belong to a same class by a single representative. This group of prototypes is built by using the NCN, instead of the NN, of a given sample because in general, those cover a bigger region.

From the experiments carried out, it seems that *Reconsistent* and *MaxNCN* provide a well balanced trade-off between accuracy and TS size reduction rate.

References

1. D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.
2. M.C. Ainslie and J.S. Sánchez. Space partitioning for instance reduction in lazy learning algorithms. In *2nd Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning*, pages 13–18, 2002.
3. C.L. Chang. Finding prototypes for nearest neighbor classifiers. *IEEE Trans. on Computers*, 23:1179–1184, 1974.
4. A. Charnes, W. Cooper, and E. Rhodes. Measuring the efficiency of decision making units. *European Journal of Operational Research*, 2:429–444, 1978.
5. B.B. Chaudhuri. A new definition of neighbourhood of a point in multi-dimensional space. *Pattern Recognition Letters*, 17:11–17, 1996.
6. C.H. Chen and A. Józwik. A sample set condensation algorithm for the class sensitive artificial neural network. *Pattern Recognition Letters*, 17:819–823, 1996.
7. B.V. Dasarathy. *Nearest neighbor (NN) norms: NN pattern classification techniques.* IEEE Computer Society Press, Los Alamitos, CA, 1990.
8. B.V. Dasarathy. Minimal consistent subset (MCS) identification for optimal nearest neighbor decision systems design. *IEEE Trans. on Systems, Man, and Cybernetics*, 24:511–517, 1994.
9. P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice Hall, Englewood Cliffs, NJ, 1982.
10. P. Hart. The condensed nearest neighbor rule. *IEEE Trans. on Information Theory*, 14:505–516, 1968.
11. C.J. Merz and P.M. Murphy. *UCI Repository of Machine Learning Databases.* Dept. of Information and Computer Science, U. of California, Irvine, CA, 1998.
12. J.S. Sánchez, F. Pla, and F.J. Ferri. On the use of neighbourhood-based non-parametric classifiers. *Pattern Recognition Letters*, 18:1179–1186, 1997.
13. I. Tomek. Two modifications of CNN. *IEEE Transactions on Systems, Man and Cybernetics, SMC-6*, pages 769–772, 1976.
14. G.T. Toussaint, B.K. Bhattacharya, and R.S. Poulsen. The application of Voronoi diagrams to nonparametric decision rules. In *Computer Science and Statistics: The Interface.* L. Billard, Elsevier Science, North-Holland, Amsterdam, 1985.
15. D.L. Wilson. Asymptotic properties of nearest neighbor rules using edited data sets. *IEEE Trans. on Systems, Man and Cybernetics*, 2:408–421, 1972.
16. D.R. Wilson and T.R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38:257–286, 2000.