A lazy learning approach for building classification models

Inés M. Galván, José M. Valls, Miguel García and Pedro Isasi

Carlos III University - Computer Science Department,
Avenida de la Universidad, 30 - 28911 Leganés (Madrid), Spain
igalvan@inf.uc3m.es

Abstract

In this paper we propose a lazy learning strategy for building classification learning models. Instead of learning the models with the whole training data set before observing the new instance, a selection of patterns is made depending on the new query received and a classification model is learnt with those selected patterns. The selection of patterns is not homogeneous, in the sense that the number of selected patterns depends on the position of the query instance in the input space. That selection is made using a weighting function in order to give more importance to the training patterns that are more similar to the query instance. Our intention is to provide a lazy learning mechanism suited to any machine learning classification algorithm. For this reason, we study two different methods to avoid fixing any parameter. Experimental results show that classification rates of traditional machine learning algorithms based on trees, rules or functions can be improved when they are learnt with the lazy learning approach proposed.

keywords: Lazy Learning, Classification Models, Pattern Selection

1 Introduction

Most of the machine learning algorithms (MLAs) -based on trees, rules, functions, etc.- are eager learning methods, in the sense that the generalization is carried out beyond the training data before observing the new instance. This is, first a model is built using the complete training data set and, afterwards, this model is used to classify the test instances.

When the training data are not evenly distributed in the input space, these global learning methods could be affected by a decrease of their generalization capabilities. Also, a model built up with the complete training data might not provide the most appropriate performance for all test instances because, intuitively, in the machine learning context, models try to extract the general

properties of data and not the individual ones. The reason of this behavior is that eager methods, in general, try to minimize the global error and this might not be the most appropriate for certain regions of the input space. Some times, this might be an interesting property, but other times this behavior could affect negatively to the generalization capability of machine learning classification models.

Local learning methods are an alternative approach [Atkenson et al., 1997], [Wettschereck et al., 1997]. They select, from the whole examples set, those patterns that are considered more appropriate for the learning task. The selection is made for each new query test presented to the system, by means of some kind of similarity measurement to that pattern. Those local methods are usually known as lazy learning or instance-based learning algorithms [Aha et al., 1991].

The typical example of lazy systems is the k-NN algorithm [Dasarathy, 1991]. In this case, the selected learning patterns are the k closest ones to the test instance by some distance metric, usually the Euclidean distance. The classification of the new query instance is just the most common class among the k selected examples. Other lazy approaches appear in the literature. For instance, Bottou and Vapnik [Bottou and Vapnik, 1992] proposed to build local linear neural networks models for each query pattern, selecting the k closest examples from the training set. In [Zhu and Yang, 2008] the authors propose a lazy bagging approach for classification. They use the k nearest neighbor from the training set and use the discovered k-NN to build bootstrap bags for bagging prediction. Also, the authors introduce a sampling-entropy-based approach to determine automatically the value of k. There are situations in which lazy approaches might get better generalization capability than eager models. However, the idea of selecting the k nearest patterns is based on the assumption that all the test patterns have the same structure and need the same selection procedure. This assumption could be invalid because the input space is neither isotropic nor homogeneous and has irrelevant and non-homogeneous features.

In [Galvan, 2001] and [Valls, 2007] we showed that the generalization capability of artificial neural networks can be improved when a lazy approach is used. Instead of training the neural networks with all the available data, they are trained with a selection of examples when a new instance is received. Thus, for each query instance a local non-linear neural model is built. To do this, the method detects how many training patterns are needed and weights them according to their similarity to the test instance. The proposed lazy method helped to improve the performance of both multilayer and radial basis neural networks.

In the current work, the goal is to extend the basic ideas of the lazy strategy already studied for neural networks [Galvan, 2001, Valls, 2007] for any classification machine learning algorithms. Given a MLA, called base algorithm, for each query instance a local model is built with the base algorithm using a subset of the whole training dataset. The subset of relevant patterns is not homogeneous and it is obtained using a weighting function, in order to give more importance to the training examples that are more similar to the query instance. A weighting function should assign high weights to the closest training examples to the

new query instance received. It reaches the maximum value when the distance to the query is null, decreasing the value smoothly as this distance increases. The number of retrieved patterns is given by that function and will depend on the new query point location in the input space. We have chosen a simple and parameter-free function: the inverse function.

In our former works related to Radial Basis Neural Networks [Valls, 2007] a parameter called radius is used to control the number of selected patterns. That radius allows to define a hyper-sphere centered in the new query instance in a way that all the training patterns inside its surface are selected to build up the local models with which the query instance will be predicted. Our studies showed that the radius is not a crucial parameter of the lazy radial basis neural networks because the results are very similar in a wide radius interval.

However, when the lazy strategy is applied to classification MLAs, the radius parameter becomes a crucial factor in the classifiers behaviour. In [Galvan, 2009] was concluded that it is not possible to fix a radius parameter appropriate for any MLA and any dataset. Each MLA might require a different number of training examples and therefore a different radius value, due to the different paradigms these methods are based on. Also the specific characteristics of different classification datasets could need different radius values. In some cases, for a fixed classifier and dataset, the classification rate might not be very dependent of the radius. However, our intention is to provide a lazy learning mechanism suited to any MLA. For this reason, in this work we study two different ways (two different methods) to avoid fixing any parameter. In the first method, the selection of training patterns only depends on the training instance weighting value, given by the inverse function. We do this in the simplest way, taking the integer part of this real value. This integer number will indicate the number of times the training instance will be included in the training set. In the second method, a procedure to automatically determine the radius parameter is proposed. Both alternatives will be explained in detail in section 2.

As we have mentioned before, this lazy approach can be applied to any MLA. In our experiments, it is applied to classification algorithms based on different paradigms, specifically C4.5, PART, Support Vector Machine and NaiveBayes algorithms. We compare both eager and lazy approaches for different MLAs. A large collection of benchmark learning problems taken form the UCI repository is used to test the method. The results show that the lazy approach can reach better generalization properties than eager or traditional methods. The aim of this work is to offer a lazy strategy that can be applied to any classifier, improving the accuracy they reach when used in the eager or traditional way.

This paper is structured as follows. Section 2 describes the method including the alternatives mentioned above. Section 3 describes the experimental validation and finally, Section 4 summarizes what has been achieved and proposes new avenues of research.

2 Lazy Learning Procedure

The general idea consists on learning a classification model for each query instance using only a selection of training patterns, those patterns close to the query instance, in terms of the Euclidean distance. A key issue of this method is to weight the examples in relation to their distance to the query instance in such a way that the closest examples have the highest weight. Thus, a weighting measure must be associated to each example. In order to use standard MLA, we replicate the examples as a way of weighting them. Therefore, the selected examples are included one or more times in the resulting training subset and the MLA is learnt with the most useful information, discarding those patterns that do not provide any knowledge.

The weight associated to each example is calculated using a kernel function which must reach its maximum value when the distance to the query point is null and must decrease smoothly as this distance increases. Although there are many functions that fulfill the above conditions, we have chosen the inverse function because is simple, intuitive and has no parameters.

Although all the training examples have a weight, not all of them must be selected. We have considered two ways of selecting the examples:

- The selection criterium depends directly on the examples weight value. In this case, we do it in the simplest way: The integer part of the weight value indicates how many times the example is replicated in the training subset.
- The selection criterium depends on a new parameter, independent of the weight value. This parameter is the radius of a sphere centered in the testing pattern. Only the examples situated into the sphere will be selected.

In the following paragraphs, we describe the general procedure to select the training subset.

Let us consider $\mathbf{q}$ an arbitrary testing pattern described by a n-dimensional vector. Let $X = \{(\mathbf{x_k}, y_k), k = 1, ..., N\}$ be the whole available training data set, where $\mathbf{x_k}$ are the input attributes and y_k the corresponding class. For each new pattern $\mathbf{q}$, the steps are the following:

First, the Euclidean distance (d_k) between each training example $\mathbf{x}_k$ and $\mathbf{q}$ is evaluated. In order to make the method independent on the distances magnitude, relative values are used. Thus, a relative distance, d_{rk} is calculated for each training pattern: $d_{rk} = d_k/d_{max}$, where d_{max} is the distance from the query to the furthest training pattern.

Next, the inverse function is used to calculate a weight for each training pattern. Thus, the weight $K(x_k)$, is the inverse of the relative distance d_{rk}:

$$K(x_k) = \frac{1}{d_{rk}}; \quad k = 1 \dots N \tag{1}$$

These values $K(x_k)$ are normalized in such a way that the sum of them equals the number of training patterns in X, this is:

$$K_N(x_k) = V \cdot K(x_k) \tag{2}$$

$$\text{where } V = \frac{N}{\sum_{k=1}^{N} K(x_k)} \tag{3}$$

At this point every training instance has a real weight value associated. Now, the training subset, named $X_{\mathbf{q}}$, associated to the testing instance $\mathbf{q}$ must be built.

As we said before, we want to study two different approaches to build X_q, depending on the criterium used to select the training examples:

- **Integer Part Approach** The selection criterium only depends on the normalized weight values ($K_N(x_k)$) that will be used to indicate how many times the training pattern (x_k, y_k) is repeated into the new training subset. Hence, they must be transformed into natural numbers. The most intuitive way to perform that transformation is to take the integer part of the weight value ($K_N(x_k)$). Thus, the pattern (x_k, y_k) is replicated n_k times in X_q, where n_k is calculated as: $n_k = \text{Int}(K_N(x_k))$.

- **Radius Approach** The selection criterium depends on a new parameter named radius (r). Depending on the dataset, the integer approach could select too many patterns, mainly in classification problems where a big amount of data are available. With the radius approach, the idea is to select only the training examples whose relative distance to the testing pattern, d_{rk}, is lower than r. In other words, only the training patterns situated into a sphere centered in the testing pattern $\mathbf{q}$ and whose radius is r, will be selected. The number of times, n_k, that the selected training patterns are replicated into the subset X_q depends on the weight values.

  Thus, the following rule is used to generate n_k:

$$\begin{array}{lll} \text{if} & d_{rk} < r & \text{then} \\ & n_k = int(K_N(x_k)) + 1 & \\ \text{else} & & \\ & n_k = 0 & \end{array} \tag{4}$$

  where $int(K_N(x_k)) + 1$ is the nearest integer greater than $K_N(x_k)$.

  When this approach is used, it is necessary to set the radius to an appropriate value, using the information of the training set. In the next subsection, we explain how the radius is automatically set.

Once the new subset X_q is built using either of both approaches, the base MLA is trained with it. Thus, a local model will be built in order to predict the testing pattern class.

In order to apply the lazy strategy explained before for any MLA, a feature must be taken into account: when the testing pattern is located into a region

of the input space where the examples are scarce, it might happen that no training examples are selected. This is a general fact that can happen with both approaches (Integer and Radius approaches). When this situation occurs, an alternative way to select the training patterns must be taken. In our work, if the subset X_q associated to a query $\mathbf{q}$ is empty, then the whole training set is used to build the model (more precisely: if X_q is empty, then $X_q \leftarrow X$). Thus, the method behaves like a global method when no training patterns are near the query instance. In addition, this transition form local to global approach is made automatically by means of the Kernel function.

2.1 Radius value determination

When a test set is to be classified by the lazy radius approach, a radius value must be fixed and it must be automatically calculated using the training set. As we said before, the appropriate value could depend on the base MLA used to build the classification model and also on the data set. Obviously, test instances can not be used to decide the most appropriate radius value for each MLA (their class value is not supposed to be available).

A n-fold cross validation procedure over the training set, varying the radius value, might be a possible way to determine the most appropriate radius. The training set would be divided into n folds and the lazy strategy explained before would be applied using the n folds as test sets, as it is usually done for this procedure. Other similar way would consist on using the leave-one-out technique over the training data set. In this case, each training pattern would be considered a test instance while the rest of patters would make up the training set.

Very often, classification data sets have few available training instances. If the radius is determined using n-fold cross validation over the training set, the success classification rate might be biased by the small size of the resulting training sets. For this reason, we believe more convenient to use the leave-one-out technique.

Hence, if S_x is the training set of instances, the mechanism to determine the radius value is the following:

- A radius value is fixed.
  - The lazy strategy based on the radius approach explained above is applied to the set S_x following a leave-one-out technique.
  - The test classification rate for this radius value is obtained.
- The radius value is varied and the procedure is repeated.
- The radius value with the best classification rate is returned.

In domains where the number of training instances is very large, we use a reduced subset of data randomly selected (for instance 50% or 20%), due to the high computational cost involved. Preliminary experiments show that using a

reduced subset is similar to using the whole training data set, when the number of instances is large enough.

3 Experimental Validation

3.1 Experimental Setup

In order to validate the proposed lazy learning approach, we have performed our experiments on a collection of machine learning datasets available from the UCI Machine Learning Repository [1]. All of them are classification domains and have numerical attributes, since our method uses Euclidean distances between patterns, although discrete attributes could also be used with the appropriate distance. A summary of some of the properties of these datasets is given in Table 1.

Table 1: Datasets Description

Name	Abbreviation	Instances	Attributes	Classes
Balance Scale	Balance	625	4	3
Bupa (Liver Disorders)	Bupa	345	6	2
Car Evaluation	Car	1728	6	4
Glass	Glass	214	9	6
Ionosphera	Ionos	351	34	2
Iris	Iris	150	4	3
Thyroid Gland	NewThyoid	215	5	3
Pima Indians Diabetes	Diabetes	768	7	2
Segmentation	Segmt	2310	19	7
Sonar	Sonar	208	60	2
Vehicle	Vehicle	846	18	4
Wine	Wine	178	13	3
Letter Recognition	Letter	16000-4000	16	26
Pen-Based Recog. of Handwritten Digits	PenDigits	7494-3498	16	10
Statlog (Landsat Satellite)	SatImage	4435-2000	36	6

Some datasets (Letter, PenDigits and SatImage) are provided with a test set. For the rest of domains, with no test set available, we perform 5 runs using 10-fold cross validation, which involves a total of 50 runs. In all cases, the attributes values have been normalized to the $[0, 1]$ interval.

The lazy approach presented before can be applied to any MLA. In this work, with the aim of representing a wide range of paradigms, we have chosen as the base MLA the following algorithms:

- A classification algorithm based on trees, C4.5 [Quinlan, 1993].
- An algorithm based on rules, PART [Quinlan, 1993]

[1] http://archive.ics.uci.edu/ml/

- An algorithm based on functions approximations, Support Vector Machines [Vapnik, 1998].
- An algorithm based on probabilities, NaiveBayes [Langley et al., 1992].

The experiments were performed using the WEKA software package [2] which includes implementations of the classifiers mentioned before: J48 (a variant of C4.5), PART, SMO (an implementation of SVM) and NaiveBayes algorithm. The results for eager or traditional versions of MLAs are obtained directly with WEKA using for each classifier the default parameters provided by the tool. Lets remind that the eager versions of the MLAs are the standard way of using the algorithms, that is, a global classification model is built using the complete training dataset. After the learning or training phase, the test instances are classified by the model.

We have modified the WEKA Software to integrate the lazy approach studied in this work. Both lazy methods or versions have been implemented: the integer part and the automatic radius method. Thus, the comparison of eager and lazy versions is possible because the implementation and parameters of the base algorithms are identical in both eager and lazy approaches. Also, the different data folds are the same for all the algorithms.

3.2 Experimental Results

The experimental results using both versions of the lazy approach (Integer Part and Automatic Radius) and using four different base algorithms (J48, Part, SMO, NaiveBayes, respectively) are shown in Tables 2, 3, 4, 5. These tables also include the performance of the eager versions of the base algorithms.

For the domains with an unique dataset available, where 5 runs of a 10-fold crossvalidation procedure is performed, we use two tailed t-test with $alpha = 0.05$ to evaluate the comparisons significance. Tables show the average success rate for the test data and the significance test result when comparing with the eager version of the algorithm (first column). For PenDigits, SatImage and Letters domains, provided with training and test sets, the significance tests have no meaning, so they have not been performed. The notation used is the following: "(+)" means that the average value is significantly better than the result provided by the eager approach; "(=)" indicates that the difference is not significant; and, "(-)" means that the lazy approach is significantly worse than the eager approach.

Tables 2 and 3 show that in most datasets both lazy approaches (Integer Part and Automatic Radius) improve the performance of eager versions of SMO and Naive Bayes algorithms. Both lazy approaches for SMO are significantly better than eager version in 12 out of 15 domains. Similar behavior has the Naive Bayes algorithm where in 10 domains out of 15 the integer part method behaves better than the eager one, and in 9 out of 15 the automatic radius method outperforms the eager one. It is also interesting to remark that for

[2] http://www.cs.waikato.ac.nz/ml/weka/

Table 2: Classification success rate: eager and lazy approaches for SMO

Dataset	Eager Approach	Lazy Approach Integer Part	Lazy Approach Automatic Radius
Balance	87.77	90.78 (+)	90.04 (+)
Bupa	58.04	70.33 (+)	68.19 (+)
Car	93.68	97.06 (+)	98.68 (+)
Glass	57.74	72.5 (+)	67.29 (+)
Ionos	88.1	94.14 (+)	91.18 (+)
Iris	96.67	95.73 (=)	95.8 (=)
NewThyoid	89.4	96.37 (+)	95.55 (+)
Diabetes	76.85	78.02 (=)	76.3 (=)
Segmt	92.94	96.03 (+)	97.04 (+)
Sonar	76.27	88.17 (+)	85.12 (+)
Vehicle	74.37	78.73 (+)	80.87 (+)
Wine	99	98.76 (=)	97.75 (=)
Letter	82.15	91.25	96.63
PenDigits	94.94	97.85	98.12
SatImage	85.21	89.37	89.07

Table 3: Classification success rate: eager and lazy approaches for NaiveBayes

Dataset	Eager Approach	Lazy Approach Integer Part	Lazy Approach Automatic Radius
Balance	90.62	90.62 (=)	90.57 (=)
Bupa	55.25	68.66 (+)	66.1 (+)
Car	85.6	91.67 (+)	94.92 (+)
Glass	46.17	71.76 (+)	68.29 (+)
Ionos	82.4	91.41 (+)	92.2 (+)
Iris	95.47	96 (=)	95.33 (=)
NewThyoid	96.84	97.02 (=)	97.31 (=)
Diabetes	75.68	75.94 (=)	75 (=)
Segmt	80.25	89.98 (+)	95.44 (+)
Sonar	67.73	83.67 (+)	83.1 (+)
Vehicle	44.85	72.5 (+)	74.44 (+)
Wine	97.29	98.87 (=)	97.41 (=)
Letter	65.05	80.03	96.7
PenDigits	82.11	91.13	97.01
SatImage	79.8	84.76	84.86

Table 4: Classification success rate: eager and lazy approaches for J48

Dataset	Eager Approach	Lazy Approach Integer Part	Lazy Approach Automatic Radius
Balance	77.89	80.03 (+)	84.95 (+)
Bupa	66.38	62.47 (=)	65.35 (=)
Car	92.34	95.88 (+)	97.49 (+)
Glass	72.99	73.24 (=)	73.14 (=)
Ionos	89.63	90.44 (=)	90.71 (=)
Iris	94.8	94.53 (=)	95.07 (=)
NewThyoid	92.67	91.67 (=)	93.03 (=)
Diabetes	74.17	72.5 (=)	73.31 (=)
Segmt	96.78	97.23 (=)	97.07 (=)
Sonar	73.86	78.07 (=)	79.06 (=)
Vehicle	71.87	71.89 (=)	72.36 (=)
Wine	93.59	94.59 (=)	94.04 (=)
Letter	87.7	89.65	94.2
PenDigits	92.05	94.28	96.25
SatImage	83.36	83.41	84.26

Table 5: Classification success rate: eager and lazy approaches for PART

Dataset	Eager Approach	Lazy Approach Integer Part	Lazy Approach Automatic Radius
Balance	82.88	83.03 (=)	84.57 (=)
Bupa	65.67	63.72 (=)	63.1 (=)
Car	95.68	97.66 (+)	97.86 (+)
Glass	73.61	73.99 (=)	72.58 (=)
Ionos	90.6	91.3 (=)	90.15 (=)
Iris	94.4	94.53 (=)	95.2 (=)
NewThyoid	94.8	92.59 (=)	95.16 (=)
Diabetes	72.68	73.12 (=)	71.41 (=)
Segmt	96.61	97.08 (=)	97.05 (=)
Sonar	78.76	77.76 (=)	80.49 (=)
Vehicle	72.41	73.03 (=)	72.2 (=)
Wine	92.58	94.03 (=)	95.29 (=)
Letter	88.58	90.68	93.75
PenDigits	93.65	94.39	96.19
SatImage	82.76	83.46	85.16

some data sets the improvement over the eager classification rate is very large when one of the lazy methods is used. For instance, when Naive Bayes is the base algorithm, the results for Bupa domain improve around 10%, 23% for Glass, 10% for Segmentation, 15% for Sonar and 27% for Vehicle. When the base algorithm is SMO, a similar situation occurs. For Bupa the improvement over the eager version of the algorithm is around 10%, for NewThyoid 6%, for Sonar 8%, for Vehicle 4%, for Letter 9% and SatImage 4%. In some of these cases, the classification rates of eager versions of NaiveBayes and SMO are very poor, and therefore, the algorithm improvement potential is quite large.

For J48 and PART algorithms (see Tables 4 and 5), both lazy approaches overcome the eager versions of the algorithms for 5 (Balance, Car, Letter, PenDigits and SatImage) and 4 (Car, Letter, PenDigits and SatImage) datasets, respectively. In the rest of domains the lazy approaches performance is similar than eager versions, that is differences are not statistically significant. In general, it seems that lazy versions of algorithms based on trees or rules are not so efficient as lazy versions of other types of algorithms. However, for some datasets (Balance, Car, Letter and PenDigits for J48; Letter for Part) the lazy approaches can improve around or more than 5% over the eager versions of the methods.

We also want to remark that in domains where both the training and the test sets are provided (Letters, PenDigits and SatImage), the lazy versions of the four base algorithms behave better than the eager ones. This behavior is probably due to the amount of available data (much larger than in the rest of domains). When the number of training examples is very high, the use of a different local model for each test instances seems to be more appropriate that the use of a single global model for all of them.

3.3 Comparative Analysis of the two lazy approaches

Comparing both lazy approaches, from a general point of view it is not possible to conclude than one approach is better than the other. Figures 1, 2, 3, and 4 represent the differences in the classification success rate between each lazy approach (Integer Part or Automatic Radius) and the eager version for each base algorithm (Fig.1: SMO, Fig.2: Naive Bayes, Fig.3: J48 and Fig.4: PART). If those differences are positive it means that the lazy method behaves better than the eager method. If they are negative, then the lazy version is worst than the eager one. These values refer to the mean, although it has been analyzed before that negative differences are not statistically significant. We observe that both lazy approaches behave in a similar way, in the sense that if a lazy version is better than the eager one, then in most cases the other lazy version has the same tendency. The same behaviour occurs when a lazy version is equal or worst (in the classification success rate) than the eager one.

Figures 3 and 4 show that for J48 and PART algorithms the automatic radius method is usually better than the integer part method. This tendency is not so clear for SMO and Naive-Bayes algorithms (Figures 1 and 2).

We want to remark that for datasets where the amount of data is very large

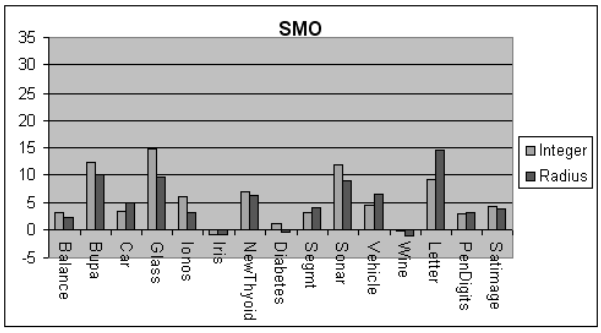

Figure 1: Lazy Approaches comparison for SMO base algorithm

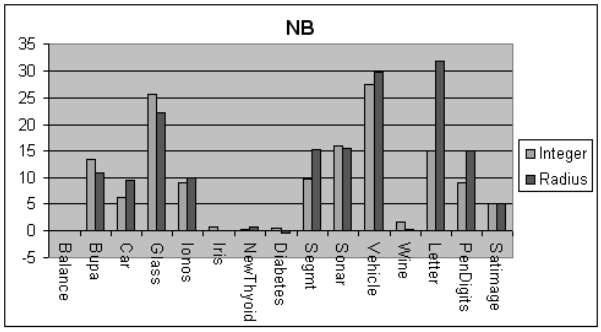

Figure 2: Lazy Approaches comparison for NB base algorithm

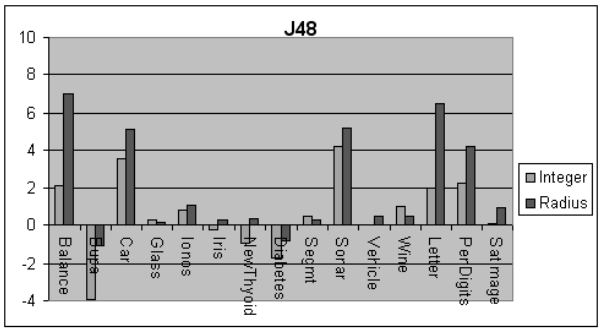

Figure 3: Lazy Approaches comparison for J48 base algorithm

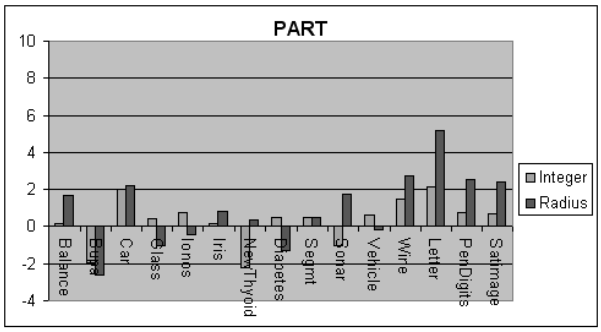

Figure 4: Lazy Approaches comparison for PART base algorithm

(Car, Letter, PenDigits and SatImage) the automatic radius method attains better results than the integer part method. This difference is very clear with PART and J48 for the four datasets. With Naive-Bayes, in four datasets the automatic radius method behaves better than the integer part one, except in SatImage where both methods behave similarly. With SMO this difference is only clear for Letter and Car domains, whereas for PenDigits and SatImage both methods are similar. The large amount of data could facilitate the search of the appropriate radius. In any case, the integer part method also behave better than the eager method, although the improvement is not that high.

Finally, and with the aim of giving a general idea about the improvement of the lazy learning approach versus the eager one for different domains, Table 6 summarizes the best results for all the datasets. The first column indicates the domain; the second one shows the best classification success rate obtained with any of the eager versions of the algorithms and the third one shows which base algorithm attained that result. The forth and fifth columns display the results corresponding to the lazy approaches: the forth column shows the best classification rate and the fifth one shows which lazy algorithm obtained that best result. The best classification rates for each domain , with statistic significance, is marked in bold. We can see that the best performance is achieved with a lazy approach of some base algorithm in 8 datasets out from the 15 datasets used for the experimental validation. They are Bupa, Car, Ionos, Sonar, Vehicle, Letter, PenDigits and SatImage. In these cases, the improvement of the lazy algorithm is 3.95%, 3%, 4.08%, 9.41%, 6.5%, 7.49%, 3.18%, 4.16%, respectively. For the rest of datasets, lazy approaches do not improve the eager algorithms, providing similar results (from a statistic points of view).

4 Conclusions

Most machine learning algorithms (MLAs) -based on trees, rules, functions, etc.- are eager learning methods; they build a model using the complete training

Table 6: Best success rate classification

Dataset	Eager Approach		Lazy Approach	
	Best Success Rate	Algorithm	Best Success Rate	Algorithm
Balance	90.62	NB	90.78	LazySMO (Integer)
Bupa	66.38	J48	**70.33**	LazySMO (Integer)
Car	95.68	PART	**98.68**	LazySMO (Radius)
Glass	73.61	PART	73.99	LazyPart (Integer)
Ionos	90.06	PART	**94.14**	LazySMO (Integer)
Iris	96.67	SMO	96	LazyNB (Integer)
NewThyoid	96.84	NB	97.31	LazyNB (Radius)
Diabetes	76.85	SMO	78.02	LazySMO (Integer)
Segmt	96.78	J48	97.23	LazyJ48 (Integer)
Sonar	78.76	PART	**88.17**	LazySMO (Integer)
Vehicle	74.37	SMO	**80.87**	LazySMO (Radius)
Wine	99.00	SMO	98.87	LazyNB (Integer)
Letter	88.58	PART	**96.70**	LazyNB (Radius)
PenDigits	94.94	SMO	**98.12**	LazySMO (Radius)
SatImage	85.21	SMO	**89.37**	LazySMO (Integer)

dataset and, afterwards, this model is used to classify the test instances. In general, these models try to extract the general properties of data and not the individual ones because they try to minimize the global error and this might not be the most appropriate for certain regions of the input space. Sometimes, this behavior could affect negatively to the generalization capability of the models. Lazy learning methods are an alternative approach. For each test instance to be classified, they select, from the whole training set, the most appropriate samples for the learning task. The selection is made by means of some kind of similarity measurement to the test pattern.

In this work we propose to apply a lazy approach to any classification machine learning algorithm. Given a MLA, called base algorithm, for each query instance a local model is built with the base algorithm using a subset of the whole training dataset. This subset of similar patterns is not homogeneous and it is obtained using the inverse function, as a parameter-free weighting function, in order to give more importance to the training examples that are more similar to the query instance. We present two different methods that avoid fixing any parameter: in the first one, the selection of training patterns only depends on the training instance weighting value, given by the inverse function. The integer part of this value indicates the number of times the training instance will be included into the training set. In the second approach, a parameter (the radius) is needed, but the method is able to automatically determine it.

In order to validate the proposed lazy learning method, we have performed our experiments on a collection of 15 machine learning datasets available from the UCI Machine Learning Repository. Both variants of the lazy approach (the integer part method and the radius method) have been compared using four different base algorithms: J48, Part, SMO, and NaiveBayes.

The results of our experiments show that the lazy approaches proposed in this work could be an easy alternative to improve the generalization of eager versions of any MLAs. For SMO and NaiveBayes, both lazy approaches improve the performance of eager versions of these algorithms in most datasets. Moreover, for some datasets, the improvement of one of the lazy methods over the eager one is quite large. For Part and J48 algorithms, lazy methods outperform the eager one in 4 and 5 datasets respectively. In the rest of domains, the performance of the lazy methods are not significatively different from the eager one. It is also possible to observe that when the amount of data is very large, the lazy versions of all the base algorithms behave significatively better than the eager ones. In these kind of domains, the use of a specific local model for each test instance seems to be more appropriate than a single global model for all the test instances.

We have compared the performance of both lazy approaches. For domains with a big amount of data, in general, the automatic radius approach behave better than the integer part one. This could be due to the fact that the large amount of data could facilitate the search of the appropriate radius.

5 Acknowledgment

This work has been funded by the Spanish Ministry of Science under contract TIN2008-06491-C04-03 (MSTAR project)

References

[Aha et al., 1991] Aha, D., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

[Atkenson et al., 1997] Atkenson, C., Moore, A., and Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, 11:11–73.

[Wettschereck et al., 1997] Wettschereck, D., Aha, D., and Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314.

[Dasarathy, 1991] Dasarathy, B. (1991). Nearest neighbour(NN) norms: NN pattern classification techniques. *IEEE Computer Society Press*.

[Bottou and Vapnik, 1992] Bottou, L. and Vapnik, V. (1992). Local learning algorithms. *Neural Computation*, 4(6):888–900.

[Zhu and Yang, 2008] Xingquan Zhu and Ying Yang (2008). A Lazy Bagging Approach to Classification. *Pattern Recognition*, 41 (10): 2980–2992.

[Galvan, 2001] I.M. Galván and P. Isasi and R. Aler and J.M. Valls (2001). A Selective Learning Method to Improve the Generalization of Multilayer Feed-

forward Neural Networks. *International Journal of Neural Systems*, 10:167-177.

[Valls, 2007] José M. Valls, Inés M. Galván and Pedro Isasi (2008). LRBNN: A Lazy RBNN Model. *AI Communications*, 20(2):71-86

[Galvan, 2009] Ineś M. Galván and José M. Valls and Nicolas Lecomte and Pedro Isasi(2009). A lazy Approach for Machine Learning Algorithms. *5th IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI 2009)*, 296: 517-522.

[Quinlan, 1993] Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory*. John Wiley and Sons.

[Langley et al., 1992] Langley, P., Iba, W., and Thompson, K. (1992). An analysis of bayesian classifiers. In *National Conference on Artificial Intelligence*.