A Lazy Approach for Machine Learning Algorithms

Inés M. Galván, José M. Valls, Nicolas Lecomte and Pedro Isasi

Abstract Most machine learning algorithms are eager methods in the sense that a model is generated with the complete training data set and, afterwards, this model is used to generalize the new test instances. In this work we study the performance of different machine learning algorithms when they are learned using a lazy approach. The idea is to build a classification model once the test instance is received and this model will only learn a selection of training patterns, the most relevant for the test instance. The method presented here incorporates a dynamic selection of training patterns using a weighting function. The lazy approach is applied to machine learning algorithms based on different paradigms and is validated in different classification domains.

1 Introduction

Lazy learning methods [1, 2, 9] defer the decision of how to generalize or classify until a new query is encountered. When the query instance is received, a set of similar related patterns is retrieved from the available training patterns set and it is used to classify the new instance. To select these similar patterns, a distance measure is used having nearby points higher relevance. Lazy methods generally work by selecting the k nearest input patterns to the query points, in terms of the Euclidean

Inés M. Galván
Universidad Carlos III, Leganés (Madrid) e-mail: igalvan@inf.uc3m.es

José M. Valls
Universidad Carlos III, Leganés (Madrid), e-mail: jvalls@inf.uc3m.es

Nicolas Lecomte
Universidad Carlos III, Leganés (Madrid) e-mail: nicolasm.lecomte@laposte.net

Pedro Isasi
Universidad Carlos III, Leganés (Madrid) e-mail: isasi@ia.uc3m.es

distance. Afterwards, the classification or prediction of the new instances is based on the selected patterns. The most popular lazy algorithm is the k-nearest neighbor method [3]. In this case, the classification of the new sample is just the most common class among the k selected examples. A variant of this methods is the weighted k-nearest neighbor [3], which consists of weighting the contribution of each of the k neighbors. Other strategy is the locally weighted linear regression [2] that constructs a linear approximation over a region around the new query instance. The regression coefficients are based on the k nearest input patterns.

Most of the machine learning algorithms (MLA) -based on trees, rules, neural networks, 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 up using the complete training data set and, afterwards, this model is used to classify the test instances. Some times, eager approximations could lead to poor generalization properties because training data are not evenly distributed in the input space. In [4, 5] the authors show that the generalization capability of artificial neural networks is improved when a lazy approach is used. Instead of using the complete training data to train the neural networks, they are trained when a new instance is received using a selection of training samples, which helps to improve the performance of the neural networks.

In this work, we propose to build up classification models using a lazy strategy instead of an eager approach, as usual. The lazy learning approach basically consists on recognizing from the whole training data set the most similar samples to each new query to be predicted. Once a subset of training patterns is selected, the classification model is learned with that subset and used to classify the new query. The subset of relevant patterns is obtained using a weighting function, the inverse function, that assigns high weights to the closest training examples to the new query instance received. The lazy approach studied in this work can be applied to any MLA. In this work, it is applied to classification algorithms based on different paradigms, specifically C4.5, PART, Support Vector Machine and NaiveBayes algorithms. Different classification domains are used to validate the method and the results show that the lazy approach can reach better generalization properties.

2 Lazy Approach for Machine Learning Algorithms

The general idea consists of learning a classification model for each query instance using only a selection of training patterns. 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. The selected examples are included one or more times in the resulting training subset.

Next, we describe the steps of the lazy approach. 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:

1. The standard Euclidean distances d_k from the pattern $\mathbf{q}$ to each input training pattern are calculated.
2. In order to make the method independent on the distances magnitude, relative distances must be 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 novel input pattern to the furthest training pattern.
3. A weighting function or kernel function is used to calculate a weight for each training pattern from its distance to the test pattern. This function is the inverse of the relative distance d_{rk}:

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

4. 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:

$$K_N(x_k) = \frac{N}{\sum_{k=1}^{N} K(x_k)} \cdot K(x_k) \tag{2}$$

5. Both the relative distance d_{rk} and the normalized weights $K_N(x_k)$ are used to decide whether the $k-th$ training pattern is selected and -in that case- how many times is included in the training subset. They are used to generate a natural number, n_k, following the next rule:

$$\begin{array}{ll} \text{if} & d_{rk} < r \qquad\qquad \text{then} \\ & n_k = int(K_N(x_k)) + 1 \\ \text{else} & n_k = 0 \end{array} \tag{3}$$

   where $int(K_N(x_k))$ is the largest integer lower than $K_N(x_k)$. r is a parameter of the method and it means the radius of a n-dimensional sphere centered at the test pattern. The idea is to select only those patterns placed into this sphere.
6. A new training subset associated to the testing pattern $\mathbf{q}$, named X_q, is built up. The $k-th$ training pattern from the original training set X is included in the new subset if it is in the sphere centered at test pattern $\mathbf{q}$ and radius r, this is $d_{rk} < r$. In addition, the $k-th$ pattern is placed n_k times randomly in the training subset X_q.
7. Finally, the MLA is trained using the new subset X_q. Thus, a local model will be built in order to predict the testing pattern class.

3 Experimental Results

In this paper we have applied the lazy proposed method to five domains from the UCI Machine Learning Repository [1]: Bupa, Diabetes, Glass, Vehicle and, Balance.

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

All of them are classification domains with numerical attributes, although discrete attributes could also be used with the appropriate distance. Also, different MLAs have been chosen as the base algorithm. Although the lazy method can be applied to any MLA, in this work we have used an algorithm based on trees, C4.5 [7]; an algorithm based on rules, PART [7]; an algorithm based on functions approximations, Support Vector Machines [8]; and an algorithm based on probabilities, NaiveBayes [6].

The experiments were performed using the WEKA software package [10] that 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.

The lazy method studied in this paper is implemented and incorporated in the WEKA Software. 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.

In all the experiments the attributes values have been normalized to the $[0,1]$ interval. For every domain and every MLA we performed 10 runs using 10-fold cross-validation, which involves a total of 100 runs. The success rate on validation data is averaged over the total number of runs.

When the lazy approach is applied, the relative radius is set as a parameter. In the cases where no training patterns are selected, due to the specific characteristics of the data space and the value of the radius, the lazy approach used the complete training data.

Table 1 displays the average success rate on validation data of the classifiers using the traditional or eager way and the lazy approach studied in this work for the different classification domains, respectively. In most domains and with most MLAs, the lazy approach is better that the eager version of the algorithms. Only, in few cases the performance of the lazy approach is similar to those provided by the eager version, but it is never worse. For instance, in Diabetes domain the performance of the lazy approach is equal than the eager one for all the classification algorithms. This also happens for some classifier in the other domains (Glass domain using J48, Vehicle Domain using Part, Balance using NaiveBayes). However, in most cases, the lazy approach provides a very important improvement.

When the performance of the MLA is poor, the lazy approach reaches more than 10% of improvement. For instance, this can be observed for the lazy version of SVM and NaiveBayes in Bupa and Glass domains, or for the lazy version of J48 in Balance domain.

Comparing both the eager and the lazy versions of all the algorithms, it is interesting to note that the best result in Bupa, Glass and Vehicle domains, is obtained by the lazy approach of one of the algorithms. In Table 1 the best classification rate for each domain is marked in bold. For the Bupa domain the best result is 68.90 %, for the Glass domain 74.20%, for Vehicle 77.78 %, all of them obtained by the lazy version of one of the algorithms. For the Diabetes and Balance domains, the results obtained by both the eager and the lazy approaches are the same.

Table 1 Classification rate: eager and lazy version of different MLAs

Domain	Algorithm	Eager Version	Lazy Version
Bupa	J48	65.84	67.43 (r=0.05)
	PART	65.25	66.81 (r=0.05)
	SVM	58.01	**68.90** (r=0.2)
	NaiveBayes	55.29	66.03 (r=0.2)
Diabetes	J48	74.68	74.68 (r=0.02)
	PART	73.05	73.05 (r=0.02)
	SVM	**76.93**	**76.97** (r=0.02)
	NaiveBayes	75.70	75.70 (r=0.02)
Glass	J48	73.61	73.79 (r=0.2)
	PART	73.32	**74.20** (r=0.2)
	SVM	57.81	70.17 (r=0.2)
	NaiveBayes	46.23	69.69 (r=0.2)
Vehicle	J48	73.61	73.79 (r=0.2)
	PART	73.32	**74.20** (r=0.2)
	SVM	57.81	70.17 (r=0.2)
	NaiveBayes	46.23	69.69 (r=0.2)
Balance	J48	77.82	85.10 (r=0.2)
	PART	83.17	85.36 (r=0.2)
	SVM	87.62	87.70 (r=0.1)
	NaiveBayes	**90.53**	**90.53** (r=0.1)

For the lazy version of MLAs we have made experiments with different radius values for each domain and each classification algorithm. The classification rates displayed in the second column of tables 1 correspond to the radius value that provided the best performance. We have observed that each domain could need a different radius value, because it depends on how the data are distributed in the input space. We have also observed that in some domains (Diabetes, Glass and Vehicle) the most appropriate radius value is the same, independently of the MLA used as base algorithm. However, in the Bupa domain for J48 and PART the most appropriate radius is 0.05 whereas for SVM and NaiveBayes is 0.2. This also happens in the Balance domain where the best result obtained by the lazy version of J48 and PART corresponds to a radius value of 0.2; conversely, SVM and NaiveBayes algorithms need a value of 0.1 to obtain the best rate. Certainly, the radius value is a parameter of the method. Each MLA might require a different number of training examples (which implies a different radius value) due to the different paradigms these methods are based on.

4 Conclusions

Most MLAs are eager learning methods because they build a model using the whole training data set and then this model is used to classify all the new query instances. The built model is completely independent of the new query instances. Lazy meth-

ods work in a different way: when a new query instance needs to be classified, a set of similar patterns from the available patterns set is selected. The selected patterns are used to classify the new instance. Sometimes, eager approximations could lead to poor generalization properties because training data are not evenly distributed in the input space and a lazy approach could improve the generalization results.

In this paper, we present a lazy method that can be applied to any MLA. In order to validate the method, we have applied it to some well-known UCI domains (Bupa, Diabetes,Glass, Vehicle and Balance Scale) using classification algorithms based on different paradigms, specifically C4.5, PART, Support Vector Machine and NaiveBayes algorithms. The results show that a lazy approach can reach better generalization properties. It is interesting to note that the lazy approaches are never outperformed by the eager versions of the algorithms. In Bupa, Glass and Vehicle domains the best results are obtained by the lazy version of any of the algorithms. In Diabetes and Balance domains the best results are obtained by both the eager and the lazy version of a specific algorithm. In some cases, when the eager versions of the algorithms have a poor performance, the lazy versions obtain a significant improvement.

Acknowledgements This article has been financed by the Spanish founded research MEC projects OPLINK:UC3M Ref:TIN2005-08818-C04-02 and MSTAR:UC3M Ref:TIN2008-06491-C04-03.

References

1. Aha D.W., Kibler D., Albert M.: Instanced-based learning algorithms. *Machine Learning*, 6:37–66 (1991).
2. Atkeson C.G., Moore A.W., Schaal S.: Locally weighted learning. *Artificial Intelligence Review*, 11:11–73 (1997).
3. Dasarathy, B.: Nearest neighbour(NN) norms: NN pattern classification techniques. *IEEE Computer Society Press* (1991).
4. Galvan I.M., Isasi P. , Aler R., Valls, J.M.: A selective learning method to improve the generalization of multilayer feedforward neural networks. *International Journal of Neural Systems*, 11:167–157 (2001).
5. Valls J.M., Galvan I.M., Isasi P.: Lrbnn: A lazy radial basis neural network model. *Journal AI Communications*, 20(2):71–86 (2007).
6. Langley P., Iba W., Thompson, K.: An analysis of bayesian classifiers. In *National Conference on Artificial Intelligence* (1992).
7. Quinlan R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo (1993).
8. Vapnik V.: *Statistical Learning Theory*. John Wiley and Sons (1998).
9. Wettschereck D., Aha D.W., Mohri T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314 (1997).
10. Witten I., Frank E.: *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann (2005)