IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, September 2010
ISSN (Online): 1694-0814
www.IJCSI.org

A Lazy Ensemble Learning Method to Classification

Haleh Homayouni[1], Sattar Hashemi[2] and Ali Hamzeh[3]

[1] Computer Science and IT Department, Shiraz University, Shiraz, Iran

[2] Computer Science and IT Department, Shiraz University, Shiraz, Iran

[3] Computer Science and IT Department, Shiraz University, Shiraz, Iran

Abstract

Depending on how a learner reacts to the test instances, supervised learning divided into *eager learning* and *lazy learning*. Lazy learners endeavor to find local optimal solutions for each particular test instance. Many approaches for constructing lazy learning have been developed, one of the successful one is to incorporate lazy learning with ensemble classification. Almost all lazy learning schemes are suffering from reduction in classifier diversity. Diversity among the members of a team of classifiers is deemed to be a key issue in classifier combination. In this paper we proposed a Lazy Stacking approach to classification, named *LS*. To keep the diversity of classifiers at a desire level, LS utilizes different learning schemes to build the base classifiers of ensemble. To investigate LS's performance, we compare LS against four rival algorithms on a large suite of 12 real-world benchmark datasets. Empirical results confirm that LS can statistically significantly outperform alternative methods in terms of classification accuracy.

Keywords: *Classification, diversity, classifier ensemble, stacking, lazy learning***.**

1. Introduction

Machine learning is a domain intensively developed in last decades. One of its main sub-domain is *supervised learning* that form decision theories or functions to accurately assign unlabeled (test) instances into different pre-defined classes.

Depending on how a learner reacts to the test instances, we have *eager learning* and *lazy learning* (Friedman et al. 1996). Eager learning methods construct a general, explicit description of the target function when training examples are provided. Instance-based learning methods simply store the training examples, and generalizing beyond these examples is postponed until a new instance must be classified. Each time a new query instance is encountered, its relationship to the previously stored examples is examined in order to assign a label to the new instance. (Aha et al. 1991).

Many approaches for constructing lazy learning have been developed. One of the most successful uses of lazy learning is in ensemble classifiers. Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a vote of their predictions. A set of classifiers with similar training performances may have different generalization performances, combining outputs of several classifiers reduces the risk of selecting a poorly performing classifier. It has been discovered that a classifier ensemble can often outperform a single classifier. A large body of research exists on classifier ensembles and why ensemble techniques are effective. (Bay 1998, Breiman 1996a, Freund & Schapire 1997, Kittler et al., 1998, Mesterharm 2003).

One of the best approaches that work under instance-base-learning in ensemble classifiers is lazy bagging (LB), *by* Xingquan Zhu et.al(2008), that builds bootstrap replicate bags based on the characteristics of test instances.

Although lazy bagging has a great success in getting more accurate classifier, diversity would be reduce, because lazy learning suffer from reducing *diversity*. In order to make the ensemble more effective, there should be some sort of *diversity* between the classifiers (Kuncheva, 2005). Two classifiers are diverse if they make different errors on new data points.

Stacking is a simple yet useful approach to this problem in order to achieve classifier diversity (Wolpert 1992). Under stacking we use different individual classifier in each bag for classifying the test instance. Stacking learns a function that combines the predictions of the individual classifiers. So different "type" of base classifiers is used in order to get more accuracy in ensembles (Seewald 2003).

In this paper, we argue that a previously successful instance-based ensemble learning model (LB) can be improved. Lazy learners are suffering from reducing diversity, because they form a decision theory that is especially tailored for the test instance. The above observations motivate our research to find a method that annihilates reducing diversity. Stacking is an ensemble that uses different "type" of base classifiers for labeling new instance. So we have expected that by using stacking along with lazy learners, we can provide the desire *diversity.*

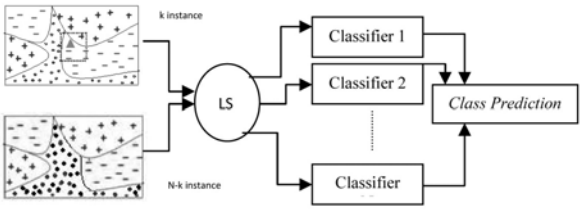

Fig. 1 *LS Diagram*; LS waits until a test instance arrived and make a set with k instances out of the NN subset and also aNother set with N− k instances out of the original dataset. Afterwards the class label of the test instance is defined by the majority vote from the output of the base learners.

The idea is that for each test instance x_i, that we add a small number of its *k*NN into the bootstrap bags, from which the base classifiers are trained. Different "type" of base classifiers for labeling new instance is used in ensemble stacking. We name this method lazy stacking (LS), a method towards more diverse ensemble classifier. By doing so, we expect to increase the base classifiers' classification diversity, leading to more accurate classification of x_i.

2. Related work

Friedman et al. (1996) proposed a lazy decision tree algorithm which built a decision tree for each test instance, and their results indicated that lazy decision trees performed better than traditional C4.5 decision trees on average, and most importantly, significant improvements could be observed occasionally. Friedman et al. (1996) further concluded "building a single classifier that is good for all predictions may not take advantage of special characteristics of the given test instance". In short, while eager learners try to build an optimal theory for all test instances, lazy learners endeavor to finding local optimal solutions for each particular test instance.

K-nearest neighbors algorithm is a key element in lazy learning. The kNN is one of the most thoroughly analyzed algorithms in machine learning, due in part to its age and in part to its simplicity. Cover and Hart (1967) present early theoretical results, and Duda and Hart (1973) provide a good overview.

One of the most successful uses of lazy learning is in ensemble classifiers (ZeNobi. 2002). The main idea of an ensemble methodology is to combine a set of classifiers in order to obtain a better composite global classifier. Ensemble methodology imitates our second nature to seek several opinions before making any crucial decision.

One of the successful method of lazy learning in ensemble classifiers is *lazy bagging* (LB)Xingquan Zhu &Ying Yang (2008), which builds bootstrap replicate bags based on the characteristics of test instances.

In order to make the ensemble more effective, there should be some sort of diversity between the classifiers (Kuncheva, 2005). In ensemble classifiers, diversity is obtained by using different types of classifiers.

Stacking is a technique whose purpose is to achieve the highest generalization accuracy (Wolpert, 1992). Stacking is usually employed to combine models built by different classifiers. Stacking performance can be improved by using output probabilities for every class label from the base-level classifiers. (Dzeroski and Zenko, 2004).

3. Lazy Stacking

In this paper, we proposed *LS*, a stacking framework with lazy local learning for building an ensemble of lazy classifiers.

Lazy Stacking applies lazy local learning to the nearest neighbors of a test instance, which produces more accurate base classifies than applying the global learner. Lazy learners are suffering from reducing diversity, because they forms a decision theory that is especially tailored for the test instance, so by choosing different classifier in stacking to the whole training set, the performance of the joint lazy and stacked learners can be increased accuracy. The increase in performance of LS can mainly attributed to the diversity of our model to be outlined in the section.

3.1 Diversity in classifiers

A commonly experienced problem with population based classification methods is the gradual decline in population diversity that tends to occur over time. Diversity among the members of a team of classifiers is deemed to be a key issue in classifier combination, and has been recognized as a very important characteristic (Cunningham & Carney, 2000; Krogh & Vedelsby, 1995; Rosen, 1996; Lam, 2000; Littlewood & Miller, 1989).

Ensembles create many classifiers, and combine their outputs to improve the performance of a single classifier. If each classifier makes different errors, so their strategic combination can reduce the total error. Therefore we need base classifiers whose decision boundaries are adequately

different from those of others, such a set of classifiers is said to be ***diverse.***

The key to the success of ensemble algorithms is that, intuitively at least, they build a set of *diverse* classifiers. The success of Stacking over Bagging is partially dependent on the diversity of the predictions made by its base classifiers [14]. To see this, consider the case when all of the base classifiers make almost the same predictions. In that case the ensemble classifier would Not perform more better than any of the base classifiers taken individually and there would be less benefit to using an ensemble classifier, but if we using different classifiers in ensembles like stacking we can get a desire accuracy. Lazy Stacking has more diversity than Lazy Bagging, so this method is a useful remedy for lack of diversity in classifier ensembles.

3.2 Algorithm of Lazy Stacking

In Lazy Stacking the learning process delayed until the arrival of a test instance, Because of its lazy learning nature. When a test instance x_i needs to be classified, LS will first try to find the *k*NN of x_i from the training set *T*, and uses the discovered *k*NN, along with the original training set *T*, to build bootstrap bags for stacking prediction. We will propose a β-*similar* concept to automatically determine the value of *K* for each dataset *T,* because *k*NN of x_i play a crucial role for LS to classify x_i.

In contrast to traditional stacking which directly samples *N* instances from a training set *T*, LS sample *K* instances from the *k*NN subset (*S*) and *N-K* instances from the original learning set (*T*). The first *N-K* instances sampled from *T* are to ensure that LS-trimmed bags function similarly to pure bootstrap bags, such that LS's base classifiers can be as independent as possible. The succeeding *K* instances from *S* are to enforce x_i's *k*NN to have a better chance to appear in each bag and thus help LS build base classifiers with less variance when classifying x_i.

Instead of directly putting all x_i's *k*NN into each bag, LS applies bootstrap sampling on x_i's *k*NN subset as well. It is expected that our procedure will ensure x_i's *k*NN have a better chance to appear in each bootstrap bag, with No decrease of the bag independency.

After the construction of each bootstrap bag B'_1, LS builds a classifier C_i from B'_1, applies C_i to classify x_i and generates a prediction $C_i(x_i)$. LS repeats the same process for *L* times, by different classifiers and eventually produces *L* predictions for x_i, $C_1(x_i)$, $C_2(x_i)$,..,$C_L(x_i)$. After that, the class *y* that wins the majority votes among the *L* base classifiers is selected as the class label for x_i.

Procedure *LazyStacking*()

Learning:

1. $K \leftarrow$ Determine the value of *K* for *T*
2. Calculate attribute weights by using Information-gain Ratio (*IR*)
3. $S \leftarrow$ Find x_i's *K* nearest neighbours from *T*
4. **For** *i* from 1 to *L*

a) $B'_i \leftarrow$ Bootstrap sampling *N-K* instances from *T*

b) $P \leftarrow$ Bootstrap sampling *K* instances from *S*

c) $B'_i \leftarrow B'_i \cup P$. Build a classifier C_i by using instances in B'_i.

Classification:

d) and apply C_i to x_i. Denoting the predicted class label by $C_i(x_i)= y_i$.

5. **End For**
6. $Y \leftarrow \arg max\ y_i$ (*Y* is the class label with majority votes.)
7. Return Y

3.3 The K Value Selection

To determine the value of k, this paper follows the approach previous studied in LB. Because kNN of x_i play a crucial role for LS to classify x_i, we will propose a β-similar concept to automatically determine the value of K for each dataset T. The large value of k leads to the increase of bag dependency and very small value of k leads to decrease in accuracy so we should find an appropriate value of k. In this paper a method is proposed for defining k automatically and uses the similarity measure between TS and LS bags that defined based on entropy. We called it β-similarity and in this paper we fixed to it to 0.99 for defining k, based on similarity measure we should use the following formula:

$$W \leq \log_4 N^{1-\beta} \quad (1)$$

Where W = K/N as the ratio between the number of KNN and the total instance number in T. To help an instance x_i find similar neighbors, we need to find the weight of each attribute so that the weighted distance function can indeed capture instances similar to x_i. For simplicity, we use Information-gain Ratio (*IR*) as a weight measure for each attribute. The attribute weights is for finding the KNN subsets i.e. attributes with larger weights have stronger effects in finding the nearest neighbors of an instance. Then

uses the Normalized information gain ratio of each attribute for it's weight. Equation 3 shows this the weighted Euclidean distance for finding kNN of an instance. $IR_0(A_i)$ is the Normalized information gain for the i'th attribute and R is the total number of attributes:

$$Dis(x_i,x_l)=\frac{1}{R}\sqrt{\sum_{i=1}^{R} IR'(A_i).(x_i^{Ai}-x_l^{Ai})} \qquad (2)$$

4. Experimental Results

To further assess the algorithm performance, we compare LS and several benchmark methods including C4.5, *k*NN, TB, and LB accuracies on 12 real-world datasets from the UCI data repository (Blake & Merz 1998). We implement C4.5, *k*NN, TB, *LB and LS* predictors by using WEKA data mining tool.

In order to measure the performance of the proposed algorithms in this work, we employed 10-time 5-fold cross validation for each dataset, and assess their performance, based on the average accuracy over 10 trials.

4.1 Description of Datasets

To compare LS other benchmark methods we use 12 real-world datasets from the UCI data repository [13]. The main characteristics of these datasets are summarized in Table 1.

4.2 Rival methods

To further assess the algorithm performance, we compare LS and several benchmark methods including C4.5, *k*NN, TB, and LB accuracies on 12 real-world datasets from the UCI data repository (Blake & Merz 1998). For LB, TB and LS, we use C4.5 unpruned decision trees (Quinlan 1993) as base classifiers because these methods prefer unstable base classifiers. *K* value is determined by fixing the β value to 0.99 for all datasets. Two single learners (the C4.5 decision tree and *k*NN) are used to offer a baseline in comparing rival algorithms.

Table 1: Experimental datasets (# of attributes includes the class label). A summary of characteristics of these data sets are showed in this table.

Dataset	*# ofClasses*	*#of Attributes*	*#of Instances*
Audiology	24	70	126
Balance	3	5	625
Bupa	2	7	345
Car	4	7	1,728
Ecoli	8	8	336
Glass	6	10	214
Hayes	3	5	132
Horse	2	23	368
Krvskp	2	37	3,196
Labor	4	17	57
Sonar	2	61	208

4.3 Classification Accuracy Comparison

Table 2 reports the accuracies of rival algorithms on each benchmark dataset. Each row in the table 2 denotes the results of one dataset, and the columns report the results from different learners. Each value in the table field gives the prediction accuracy and the standard deviation. For each learner and one single dataset, the highest accuracy value among five methods is bold-faced.

Table 2: Classification accuracy (and standard deviation) on 12 datasets selected from the UCI data repository. For each dataset, the accuracy of the method with the highest mean accuracy is marked in bold face.

Dataset	*KNN*	C4.5	TB	Lazy Bagging	Lazy Stacking
Audiology	54.42±2.47	76.15±2.04	78.54±2.71	81.75±1.98	**83.30±1.76**
Labor	84.45±2.57	78.95±3.88	84.04±4.25	86.64±3.51	**90.56±2.71**
Balance	**89.28±0.98**	65.74±0.85	74.66±0.72	77.23±1.04	86.24±0.48
Pima	75.00±1.05	73.65±1.21	75.17±0.81	76.23±0.68	**76.32±1.08**
Bupa	63.47±2.21	64.20±2.90	69.17±1.88	70.23±2.04	**70.28±1.38**
Car	78.67±0.64	91.37±0.86	92.58±0.80	93.21±0.48	**94.78±0.27**
Hayes	64.36±2.68	71.32±3.1	73.50±2.37	75.76±1.18	**81.07±2.52**
Horse	81.61±1.26	**85.12±0.67**	84.10±0.94	84.96±0.63	84.35±1.01
Glass	64.95±1.87	66.92±2.65	72.62±1.88	74.31±1.62	**74.55±1.77**
Sonar	73.18±6.75	73.08±3.63	75.91±2.21	80.05±1.78	**84.66±1.87**

Ecoli	**86.01±1.07**	82.56±1.17	83.72±1.06	83.30±1.11	85.57±0.69
Kr-vs-kp	89.64±0.15	99.29±0.12	99.36±0.08	99.69±0.08	**99.71±0.12**

The mean accuracy in 12 UCI data sets by lazy bagging classification is 81.946 and by lazy stacking is 84.282. The results in Table 2 indicate that LS can outperform LB on 12 datasets with more than 2% absolute accuracy improvement, and also the results indicate that out of the 12 datasets, LS wins other methods in 16 datasets (a probability of 9/12=75%).

3.1 Experimental Results and Statistical Tests

Various researchers adopt different statistical and common-sense techniques to decide whether the differences between the algorithms are real or random. In this section we examine some kNown statistical tests, and study the performance for our algorithms.
We further proceed to compare rival algorithms across multiple data sets. We deploy the following statistical tests: absolute average accuracy, t-test.

Table 3. The comparison between lazy bagging and lazy stacking, on 12 datasets from UCI data repository (datasets are ranked based on LS's *absolute average accuracy* improvement over LB in a descending order). The *p*-value indicating a statistical difference (less than the critical value 0.05) is bolded. All *p* values less than 0.001 are deNoted by <0.001.

Dataset	Lazy Bagging	Lazy Stacking	LS_LB	p-value
Balance	77.23±1.04	86.24±0.48	9.01	**<0.001**
Sonar	80.05±1.78	**84.66±1.87**	**4.61**	**<0.001**
Lymph	80.41±2.02	**85.01±1.04**	**4.60**	**<0.001**
Labor	86.64±3.51	**90.56±2.71**	**3.92**	**0.018**
Ecoli	83.30±1.11	85.57±0.69	2.27	0.048
Hayes	75.76±1.18	**81.07±2.52**	**5.31**	**0.001**
Bupa	70.23±2.04	**70.28±1.38**	**0.05**	**0.045**
Audiology	81.75±1.98	**83.30±1.76**	**1.55**	**0.014**
Glass	74.31±1.62	**74.55±1.77**	**0.24**	0.042
Pima	76.23±0.68	**76.32±1.08**	**0.09**	**0.015**
Kr-vs-kp	**99.69±0.08**	99.71±0.12	0.02	0.218
Horse	84.96±0.63	84.35±1.01	-0.61	**0.440**

Table 3 also reports the *p*-values between LS and LB to evaluate whether the mean accuracies of LB and LS are statistically different; a statistically different value (less than the critical value 0.05) is bolded. Take the first dataset *Balance* in Table 3 as an example. The *p*-value (denoted by <0.001) indicates that the accuracies of LS are statistically significantly higher than that of LB. We then can infer that LS is statistically significantly better than LB on *Balance*.

When comparing *LS* and *LB*, we can easily conclude that *LS* significantly outperform *LB* with improvement, and for several datasets, the accuracies of *LS* are more. This asserts that for learners, a stacking framework based on lazy local learning is better design than LB statistically.

Overall in table 3, LS outperforms LB on 9 datasets, of which the results on 9 datasets are statistically significant. On the other hand, LB outperforms LS on 3 datasets. LS's wins (9) versus losses (3) compared with LB is also statistically significant. The results indicate that out of the 12 datasets, LS wins other methods in 9 datasets (a probability of 9/12=75%).

As a result when comparing L*S* to other classifier, we can find that *LS in most datasets* can outperform (or perform as good as) other classifier. Meanwhile, we also Notice that for several datasets, the accuracies of L*S are* identical to that of the other classifier and simple bootstrap sampling.

5. Conclusions & future work

In this paper, we proposed *LS*, a stacking framework with lazy local learning for building a classifier ensemble learner. The method is a combination of lazy learning and ensemble classifiers in order to get more accurate classifier.
As a future work we plan to extend the current work to combine Bagging and Stacking in lazy learners in order to better classifier.

Acknowledgments

This work was supported by the Iran Tele Communication Research Center.

References

[1] D. Aha, D. Kibler, and M.Albert, "*Instance-based learning algorit*hms''. Machine learning, Vol.6, 1991, pp. 37-66.

[2] L. Breiman, ''*Bagging predictors''*, Machine Learning, Vol.24, No.2, 1996a, pp.123-140.

[3] D. W. Aha, Editorial: ''*Lazy Learning'',* Artificial Intelligence Review, Vol.11, 1997, pp.1-5.

[4] J. Kittler, M. Hatef, R. Duin, and J. Matas, '' *On combining classifiers''*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 20, No. 3, 1988, pp. 126-239.

[5] D. Wolpert, '' *Stacked Generalization'',* Neural Networks, Vol.5, No.2, 1992, pp. 241-259.

[6] J. Friedman, R. Kohavi, and Y. Yun, '' *Lazy decision trees''*. In Proc. of the AAAI, 1996, pp.717-724. MIT Press.

[7] B. Zenkos, S. Zeroski, '' *Is Combining Classifiers with Stacking Better than Selecting the Best One?''*, Machine Learning, Vol.54, 2004, pp.255–273.

[8] X. Zhu, Y. Yang, ''*.A lazy Bagging approach to classification''*, Pattern Recognition, Vol.41, No.10, 2008, pp.2980-2992.

[9] R. Polikar, ''*Ensemble based systems in decision making*'', IEEE Circuits and Systems Magazine, Vol.6, 2006, pp.21–45.

[10] L. Kuncheva, and C. Whitaker, ''*Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy'',* Machine Learning, Vol.51, No.2, 2003, pp.181-207.

[11] R. Khoussainov, A. Heb, and N. Kushmerick, '' *Ensembles of Biased Classifiers'',* In Proc. of the 12nd ICML Conference, 2005, Vol.119**,** pp.425 – 432.

[12] R. Polikar R., ''*Ensemble Based Systems in Decision Making'',* IEEE Circuits and Systems Magazine, Vol.6, No. 3, 2006, pp. 21-45.

[13] C.L. Blake, C.J. Merz, UCI repository of machine learning database, 1992.

[14] S.A. Gilpin, D.M. Dunlavy, ''*Relationships Between Accuracy and Diversity in Heterogeneous Ensemble Classifiers'', SAND2009,694OC.* Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

H. Homayouni was born in Shiraz, Iran in 1984. She received her B.Sc. degree in Computer Engineering from meybod Islamic Azad University in 2008. She is currently an M.Sc. student in Artificial Intelligence at Shiraz University. Her research interests include supervised learning and ensemble classifiers.

S. Hashemi received the PhD degree in Computer Engineering from the Iran University of Science and Technology, in conjunction with Monash University, Australia, in 2008. He is currently a lecturer in the Electrical and Computer Engineering School, Shiraz University, Shiraz, Iran. His research interests include data stream mining, database intrusion detection, dimension reduction, and adversarial learning.

A. Hamzeh received his Ph.D. in artificial intelligence from Iran University of Science and Technology (IUST) in 2007. Since then, he has been working as assistant professor in CSE and IT Department of Shiraz University. There, he is one of the founders of local CERT center which serves as the security and protection service provider in its local area. As one of his research interests, he recently focuses on cryptography and steganography area and works as a team leader in CERT center to develop and break steganography method, especially in image spatial domain. Also, he works as one of team leaders of Soft Computing group of shiraz university working on bio-inspired optimization algorithms. He is co-author of several articles in security and optimization.