Pattern Recognition 41 (2008) 2980–2992

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A lazy bagging approach to classification[☆]

Xingquan Zhu[a,*], Ying Yang[b]

[a]Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA
[b]Faculty of Information Technology, Monash University, Melbourne, VIC 3800, Australia

ARTICLE INFO

Article history:
Received 12 May 2007
Received in revised form 5 March 2008
Accepted 8 March 2008

Keywords:
Classification
Classifier ensemble
Bagging
Lazy learning

ABSTRACT

In this paper, we propose lazy bagging (LB), which builds bootstrap replicate bags based on the characteristics of test instances. Upon receiving a test instance x_k, LB trims bootstrap bags by taking into consideration x_k's nearest neighbors in the training data. Our hypothesis is that an unlabeled instance's nearest neighbors provide valuable information to enhance local learning and generate a classifier with refined decision boundaries emphasizing the test instance's surrounding region. In particular, by taking full advantage of x_k's nearest neighbors, classifiers are able to reduce classification bias and variance when classifying x_k. As a result, LB, which is built on these classifiers, can significantly reduce classification error, compared with the traditional bagging (TB) approach. To investigate LB's performance, we first use carefully designed synthetic data sets to gain insight into why LB works and under which conditions it can outperform TB. We then test LB against four rival algorithms on a large suite of 35 real-world benchmark data sets using a variety of statistical tests. Empirical results confirm that LB can statistically significantly outperform alternative methods in terms of reducing classification error.

1. Introduction

The task of supervised classification learning is to form decision theories or functions that can be used to accurately assign unlabeled (test) instances into different pre-defined classes. Various approaches have been proposed to carry out this task. Depending on how a learner reacts to the test instances, these approaches can be categorized into two groups, *eager learning* vs. *lazy learning* [1].

Being eager, a learning algorithm generalizes a "best" decision theory during the training phase, regardless of test instances. This theory is applied to all test instances later on. Most top-down induction of decision tree (TDIDT) algorithms are eager learning algorithms, for instance C4.5 (Quinlan, 1993). In contrast, lazy learning [2] waits until the arrival of a test instance and forms a decision theory that is especially tailored for this instance. Traditional k nearest neighbor (kNN) classifiers [3] are typical lazy learners. Previous research has demonstrated that lazy learning can be more accurate than eager learning because it can customize the decision theory for each individual test instance. For example, Friedman et al. [1] 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. [1] further concluded "building a single classifier that is good for all predictions may not take advantage of special characteristics of the given test instance". Based on this conclusion, Fern and Brodley [4] proposed a boosting mechanism for lazy decision trees. 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.

For both eager and lazy learning, making accurate decision is often difficult, due to factors such as the limitations of the learning algorithms, data errors, and the complexity of the concepts underlying the data. 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 ensembling techniques are effective [5–9]. One of the most popular ensemble approaches is 'Bagging' [6]. Bagging randomly samples training instances to build multiple bootstrap bags. A classifier is trained from each bag. All these classifiers compose an ensemble that will carry out the classification task by conducting voting among its base classifiers.

For all bagging-like approaches (except for Fern and Brodley's approach [4], which was only applicable to lazy decision trees), the base classifiers are eager learners. Thus, each base learner seeks to build a globally optimal theory from a biased bootstrap bag. Because

☆ This research has been supported by the National Science Foundation of China (NSFC) under Grant no.60674109.

* Corresponding author: Tel.: +1 561 297 3168; fax: +1 561 297 2800.
E-mail addresses: xqzhu@cse.fau.edu (X. Zhu), Ying.Yang@infotech.monash.edu.au (Y. Yang)
URLs: http://www.cse.fau.edu/~xqzhu (X. Zhu), http://www.csse.monash.edu.au/~yyang/ (Y. Yang).

0031-3203/$30.00 © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.03.008

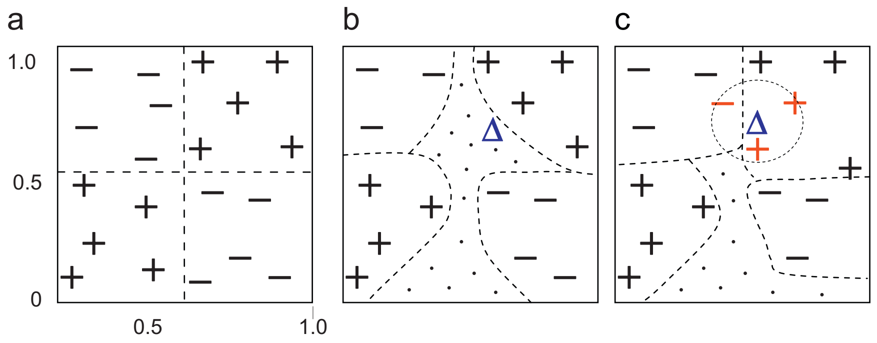

Fig. 1. A 20-instance 2-class toy data set with linear "cross" optimal decision boundaries. The horizontal and vertical axes denote the values of the 2 dimensional features. Each "+" / "−" represents a training instance belonging to a positive/negative class. (a) Original data set. (b) A bootstrap bag (dash lines denote the classifier's decision surfaces, dotted areas represent uncertain decision regions, and "Δ" denotes a test instance). Note that a bootstrap bag may contain duplicate instances. (c) Adding 3NN of "Δ" changes the decision surfaces and reduces the uncertainty for classifying "Δ".

of this, the base learners have high variance, which can reduce classification accuracy.

Consider a two-class toy data set whose opti-mal decision boundary[1] is a linear "cross" as in Fig. 1(a). Assume a learner is capable of maximizing the regions for each class of instances to form arbitrarily shaped decision boundaries, by competing with neighbors from the opposite class. The classifier's decision surface will then be able to approach the optimal decision boundary (the dash lines shown in Fig. 1(a)). Now a bagging predictor randomly samples (with replacement) 20 instances from Fig. 1(a) and builds a bootstrap replicate bag as shown in Fig. 1(b). Because of the random sampling process, instances in the bag are biased (compared with the original training set) and the decision surfaces of the classifier will change accordingly as illustrated in Fig. 1(b) where dotted areas indicate uncertain decision regions. At this point, a classifier built by Fig. 1(b) will experience difficulty in classifying the instance denoted by "Δ". If we could add 3*NN* of "Δ" into Fig. 1(b), the decision surfaces will move towards a higher decision certainty for the local regions surrounding "Δ" as in Fig. 1(c). As a result, the classifier built by Fig. 1(c) can refine its local decision boundaries for the benefit of the instance "Δ".

In short, the examples in Fig. 1 illustrate that while increasing bag independency, bootstrap sampling can introduce bias into each bag (biased towards selected instances), from which biased classifiers with high levels of variance will be constructed. Although the voting procedure may somewhat reduce the variance, it is detrimental to a bagging predictor's classification accuracy if its base learners have high variance [10,11].

The above observations motivate our research on lazy bagging (LB) [29]. The idea is that for each test instance x_k, we add a small number of its kNN into the bootstrap bags, from which the base classifiers are trained. By doing so, we expect to decrease the base classifiers' classification bias and variance, leading to more accurate classification of x_k than traditional bagging (TB) can offer. We will study LB's niche and explore conditions under which LB can outperform TB.

It is worth noting that the simple toy problem in Fig. 1 turns out to be difficult for a learner like C4.5, even with a sufficient number of training instances. In Section 3, we will demonstrate that for the same problem with 600 training instances, the classification accuracy of C4.5 is only 46.95%. A TB predictor's accuracy with 10 base classifiers is 75.83%, and the accuracy of LB with 10 base learners is 95.18%. Increasing TB's base classifier number to 200 increases TB's accuracy to 79.3%, which is still significantly lower than LB with merely 10 base classifiers.

The remainder of the paper is structured as follows: Section 2 proposes LB in detail, Section 3 studies the rationale and the niche of LB, by using the bias-variance theory and the empirical results drawn from five carefully designed synthetic data sets; Section 4 reports experimental results when comparing LB with four popular rival methods on a large suite of 35 real-world benchmark data sets and Section 5 gives concluding remarks and suggests future work. For ease of presentation, key symbols used in this paper are listed in Table 1.

Table 1
Key symbols used in the paper

Symbol	Description
LB	Lazy bagging
TB	Traditional bagging
kNN	A short hand of k nearest neighbors, it also denotes a k nearest neighbor classifier
K4.5	A lazy learner using a test instance's k nearest neighbors to build a decision tree
T	A shorthand of a data set (training set or test set)
S	A shorthand of a small instance subset
N	Number of instances in a data set
K	Number of nearest neighbors determined by LB
Y	A short hand of the whole class space
y	A short hand of a class label
x_k	A shorthand of the kth instance in the data set
y_k	A short hand of the class label of x_k
y_k^*	The prediction of the majority classifies of a classifier ensemble on y_k
B_i	An instance bag built by pure bootstrap sampling (or TB)
B_i'	An instance bag built by LB
L	Number of base classifiers of a classifier ensemble
C_i	A short hand of the ith base classifier of a classifier ensemble
i.i.d.	independent and identically distributed

2. Lazy bagging

The framework of LB is shown in Fig. 2. Because of its lazy learning nature, the learning process is delayed until the arrival of a test instance. As soon as a test instance x_k needs to be classified, LB will first try to find the kNN of x_k from the training set T, and uses the discovered kNN, along with the original training set T, to build bootstrap bags for bagging prediction. Because kNN of x_k play a crucial role for LB to classify x_k, we will propose a *β-similar* concept to automatically determine the value of K for each data set T to be detailed in Section 2.1. Finding x_k's kNN that are indeed similar to x_k

[1] In this paper, *optimal decision boundary* and *optimal decision surface* are interchangeable terms that denote the true underlying hypersurface that partitions instances into different classes. In contrast, we use *decision boundary* or *decision surface* to denote a classifier's actual hypersurface when classifying instances.

```
Input: (1) $T$: A training set with $N$ instances}
       (2) $x_k$: A test instance with unknown class label
       (3) $L$: The number of bootstrap bags
       (4) $\beta$: The $\beta$-similar value in determining the number of nearest
           neighbors for $x_k$ (Section 2.1)
Output: $y_k$: A class label for $x_k$
Procedure LazyBagging()
  1. $K \leftarrow$ Determine the value of $K$ for $T$ (Section 2.1)
  2. Calculate attribute weights (Section 2.2)
  3. $S \leftarrow$ Find $x_k$'s $K$ nearest neighbors 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. $B'_i \leftarrow B'_i \cup P$
       d. Build a classifier $C_i$ by using instances in $B'_i$, and apply $C_i$ to
          $x_k$. Denoting the predicted class label by $C_i(x_k)$.
  5. EndFor
  6. $y_k \leftarrow \arg\max_{y \in Y} \sum_{i=1; C_i(x_k)=y}^{L} 1$   ($Y$ denotes the whole class space and $y_k$
     is the class label with majority votes.)
  7. Return $y_k$
```

Fig. 2. The generic framework of lazy bagging.

is a key for LB to gain improvements. For this purpose, LB employs the information-gain ratio (*IR*) measure to discover *k*NN to be discussed in Section 2.2.

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

Instead of directly putting all x_k's *k*NN into each bag, LB applies bootstrap sampling on x_k's *k*NN subset as well. Our preliminary study indicates that any efforts in putting the same data subset into bootstrap bags will increase bag dependency, and eventually reduce the prediction accuracy. It is expected that our procedure will ensure x_k's *k*NN have a better chance to appear in each bootstrap bag, with no (or low) decrease of the bag independency.

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

The LB framework in Fig. 2 can accommodate any learning algorithms. This is essentially different from a previous boosting method which was designed for lazy decision trees only [4]. Our motivation in refining local decision boundaries for better learning shares some similarity with Triskel, a recent ensembling technique for biased classifiers [12]. Triskel forces base learners to be biased and have high precision on instances from a single class. In each iteration, it classifies and separates the "easy'' instances and then uses the ensemble members from the subsequent iterations to handle the remaining "difficult'' instances in a recursive way. By doing so, base learners are able to "converge'' to some local problems and solve the separation of a local region which may look complex globally, but simple locally.

2.1. The K value selection

The value of K decides the region surrounding a test instance x_k, from which LB can choose instances to improve the certainty of the base classifiers in classifying x_k. As shown in Fig. 2, for either $K = 0$ or $K = N$, LB will degenerate to TB, and for any other K value, a compromise between the bag dependency and the classifier certainty must be made. In this subsection, we derive a sampling-entropy-based approach to automatically determine the value of K for each data set.

Definition 1. Given a data set T with N instances, assuming $p_1, p_2, \ldots, p_N$ denote the average sampling probability for instance $x_1, x_2, \ldots, x_N$ respectively ($\sum_{n=1}^{N} p_n = 1$), then the *sampling entropy* of an *N-instance* bag B_i built by T is defined by $E(B_i) = -\sum_{n=1}^{N} p_n \log p_n$.

Lemma 1. *Given a learning set T with N instances, the sampling entropy of an N*-instance *bootstrap bag B_i built by TB is $E(B_i) = \log N$.*

Proof. TB uses i.i.d. sampling with replacement. Hence the sampling probability for each instance is $1/N$. For a bootstrap bag B_i with N instances, its sampling entropy is $E(B_i) = -(N/N)\log(1/N) = \log N$. It is obvious that a bag built by TB has the largest sampling entropy. □

Lemma 2. *Given a learning set T with N instances, the sampling entropy of an N-instance bootstrap bag B'_i built by LB with K nearest neighbors is*

$$E(B'_i) = -\frac{(N-K)^2}{N^2}\log\frac{N-K}{N^2} - \frac{(2N-K)\cdot K}{N^2}\log\frac{2N-K}{N^2}.$$

Proof. Given an instance x_i, if it does not belong to the *k*NN subset S, then in any of the first $N - K$ sampling steps (from T), x_i has $1/N$ probability to be sampled in each step. For the succeeding K sampling steps (from S), x_i's sampling probability is 0 because it does not belong to S. So the average sampling probability for x_i is $p_i = (N-K)/N^2$.

On the other hand, if an instance x_j belongs to the *k*NN subset S, then in any of the first $N-K$ sampling steps, x_j has $1/N$ probability to be sampled. In any of the succeeding K steps, x_j's sampling probability is $1/K$. So the average sampling probability for x_j is $p_j = (2N-K)/N^2$.

Obviously, there are $N - K$ and K instances in T like x_i and x_j, respectively. So the sampling entropy of B'_i is

$$E(B'_i) = -\frac{(N-K)^2}{N^2}\log\frac{N-K}{N^2} - \frac{(2N-K)\cdot K}{N^2}\log\frac{2N-K}{N^2}. \qquad \square$$

Definition 2. Given a learning set T with N instances, we say a bag B'_i built by LB is *β-similar* to a same size bag B_i built by TB, iff $E(B'_i)/E(B_i) \geqslant \beta$.

Definition 2 uses sampling entropy ratio to assess the statistical similarity between two approaches to constructing bootstrap bags. Any two bags built by TB are considered conceptually equivalent, since their sampling entropies are the same. Because LB changes the sampling probability for some instances, a bag built by LB will have less sampling entropy. The higher the β value, the more the bags built by LB are considered similar to the ones from TB.

Lemma 3. *Given a learning set T with N instances and $\omega = K/N$ as the ratio between the number of k*NN *and the total instance number in T,* *to ensure that an N instance bag of LB, B_i', is β-similar to a same size bag B_i built by TB, the value of ω must satisfy $\omega \leqslant \log_4 N^{1-\beta}$.*

Proof. According to Definition 2, to ensure that B_i' is β-similar to B_i, we must have $E(B_i')/E(B_i) \geqslant \beta$. That is,

$$\frac{-((N-K)^2/N^2)\log((N-K)/N^2)-(((2N-K)\cdot K)/N^2)\log((2N-K)/N^2)}{-\log 1/N} \geqslant \beta. \quad (1)$$

Because $\omega = K/N$, Inequity (1) can be transferred to

$$\frac{-(1-\omega)^2\log((1-\omega)/N)-(2-\omega)\cdot\omega\cdot\log((2-\omega)/N)}{\log N} \geqslant \beta. \quad (2)$$

Or equivalently,

$$\frac{-\log((((1-\omega)/N))^{(1-\omega)^2}\cdot((2-\omega)/N))^{(2-\omega)\omega})}{\log N} \geqslant \beta. \quad (3)$$

Because the number of *k*NN selected is very small compared to the total instance number N, ω is a small value. We can thus ignore its high-order values, such as ω^2. Meanwhile, because N is much larger than ω,we can simplify $(1-\omega)/N$ and $(2-\omega)/N$ to $1/N$ and $2/N$ respectively. This gives us Inequity in (4).

$$\frac{-\log((1/N)^{(1-2\omega)}\cdot(2/N)^{2\omega})}{\log N} \geqslant \beta \quad (4)$$

which is equivalent to

$$\frac{\log(N/4^{\omega})}{\log N} \geqslant \beta. \quad (5)$$

Since $\log_a b = \log_x b/\log_x a$, inequity (5) can be transferred to

$$\log_N \frac{N}{4^{\omega}} \geqslant \beta \quad (6)$$

which is equivalent to $4^{\omega} \leqslant N^{1-\beta}$. Thus finally we get $\omega \leqslant \log_4 N^{1-\beta}$. □

Lemma 3 explicitly specifies the maximal number of *k*NN for a data set *T*, if we require bags from LB to be β-similar to bags from TB. Given a specific β value, the maximal *K* value is proportional to the total instance number *N* in the learning set. Because any *K* values less than the one determined by Lemma 3 are acceptable, and LB actually prefers maximizing the number of *k*NN in bootstrap bags, LB uses the largest *K* value determined by Lemma 3.

In all our experiments, we set $\beta = 0.99$ (which, we believe, is a pretty tight value) for all data sets. This gives us a range of $\omega \in [0.033, 0.066]$ for learning sets with 100 to 10 000 training instances. That is, the number of *k*NN we selected is 3.3% to 6.6% of the total instances in the learning set (depending on the actual number of training instances).

2.2. Attribute weight and distance function

The performance of LB relies on whether the *k*NN are indeed similar to x_k or not. To help an instance x_k 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_k. For simplicity, we use *IR* as a weight measure for each attribute. Interested users can refer to Quinlan (1993) for more information about *IR*, or employ other approaches such as Relief [13] to calculate attribute weights.

After the calculation of the *IR* value for each attribute, LB normalizes all the *IR* values into range [0 1], and uses the Euclidian distance function in Eq. (7) to calculate the distance between instances, where $\Re$ denotes the total number of attributes, and $x_k^{A_i}$ denotes the value of the attribute A_i for the instance x_k. For a categorical attribute, $x_k^{A_i} - x_l^{A_i}$ equals 0 *iff* both x_k and x_l have the same value on A_i. Otherwise it equals 1:

$$\mathrm{Dis}(x_k, x_l) = \frac{1}{\Re}\sqrt{\sum_{i=1}^{\Re} IR'(A_i)\cdot(x_k^{A_i} - x_l^{A_i})^2}. \quad (7)$$

3. The rationale and the niche of LB

In order to explore why and when LB are effective in practice, we will first design five synthetic data sets, whose complexity and optimal decision boundaries we know exactly. Next, we will borrow several measures from existing research to study LB on these carefully designed benchmark data sets. The first types of measures are based on the bias-variance theory from the literature [10,14–17]; and the second types of measures are based on *Q* statistic analysis [18] for bag and instance level comparisons.

3.1. The synthetic benchmark data sets

Fig. 3 depicts all five synthetic data sets, each of which is a 2-dimensional 2-class problem with 600 instances (300 instances per class). For each data set, the left picture in Fig. 3 lists all positive and negative instances (pink vs. blue), and the right picture shows the training instances along with the biased points (dark red) in the data set (the definition of a biased point is given in Section 3.2.1).

In Fig. 3, the first three data sets ($S_{0.1}$, $S_{0.2}$ and $S_{0.5}$) are generated such that positive instances are uniformly distributed in the region between $y = \sin(x)$ and $y = \sin(x) + \delta$, and negative instances are uniformly distributed in the region between $y = \sin(x)$ and $y = \sin(x) - \delta$. Therefore, all three data sets have the same optimal decision boundaries: $y = \sin(x)$. The values of δ for $S_{0.1}$, $S_{0.2}$ and $S_{0.5}$ are set to 0.1, 0.2, and 0.5, respectively. Our objective is to investigate how LB responds to learning data sets with identical optimal decision boundaries but different data distributions.

The fourth data set, Nrm, consists of two 2-dimensional normal distributions, as defined by Eq. (8). The positive class ζ_1 (pink) and the negative class ζ_2 (blue) are given by: $\mu_1 = [\mu_{x1}, \mu_{y1}]^{\mathrm{T}} = [0, 0]^{\mathrm{T}}$, $\mu_2 = [\mu_{x2}, \mu_{y2}]^{\mathrm{T}} = [1, 0]^{\mathrm{T}}$, and

$$\Sigma_1 = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} \sigma_2^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$f_X(X|\zeta) = \frac{1}{2\pi\|\Sigma\|^{1/2}}\exp\left(-\frac{1}{2}(X-\mu)^{\mathrm{T}}\Sigma^{-1}(X-\mu)\right). \quad (8)$$

The optimal (Bayesian) decision boundary of Nrm is found by applying the likelihood test:

$$\frac{f_X(X|\zeta_1)}{f_X(X|\zeta_2)} = \frac{\sigma_2^2}{\sigma_1^2}\times\exp\left(-\frac{[(x-\mu_{x1})^2+(y-\mu_{x1})^2]}{2\sigma_1^2} + \frac{[(x-\mu_{x2})^2+(y-\mu_{x2})^2]}{2\sigma_2^2}\right). \quad (9)$$

Therefore, we can define the optimal decision boundary for Nrm data set as follows, *that is*, $f_X(X|\zeta_1) = f_X(X|\zeta_2)$:

$$\frac{1}{\sigma_2^2}[(x-\mu_{x2})^2+(y-\mu_{y2})^2] - \frac{1}{\sigma_1^2}[(x-\mu_{x1})^2+(y-\mu_{y1})^2] = 4\log\left(\frac{\sigma_1}{\sigma_2}\right). \quad (10)$$

Using simple math, we may redefine the optimal decision boundary in Eq. (10) simply as

$$\|X - X_C\|^2 = r^2 \quad (11)$$

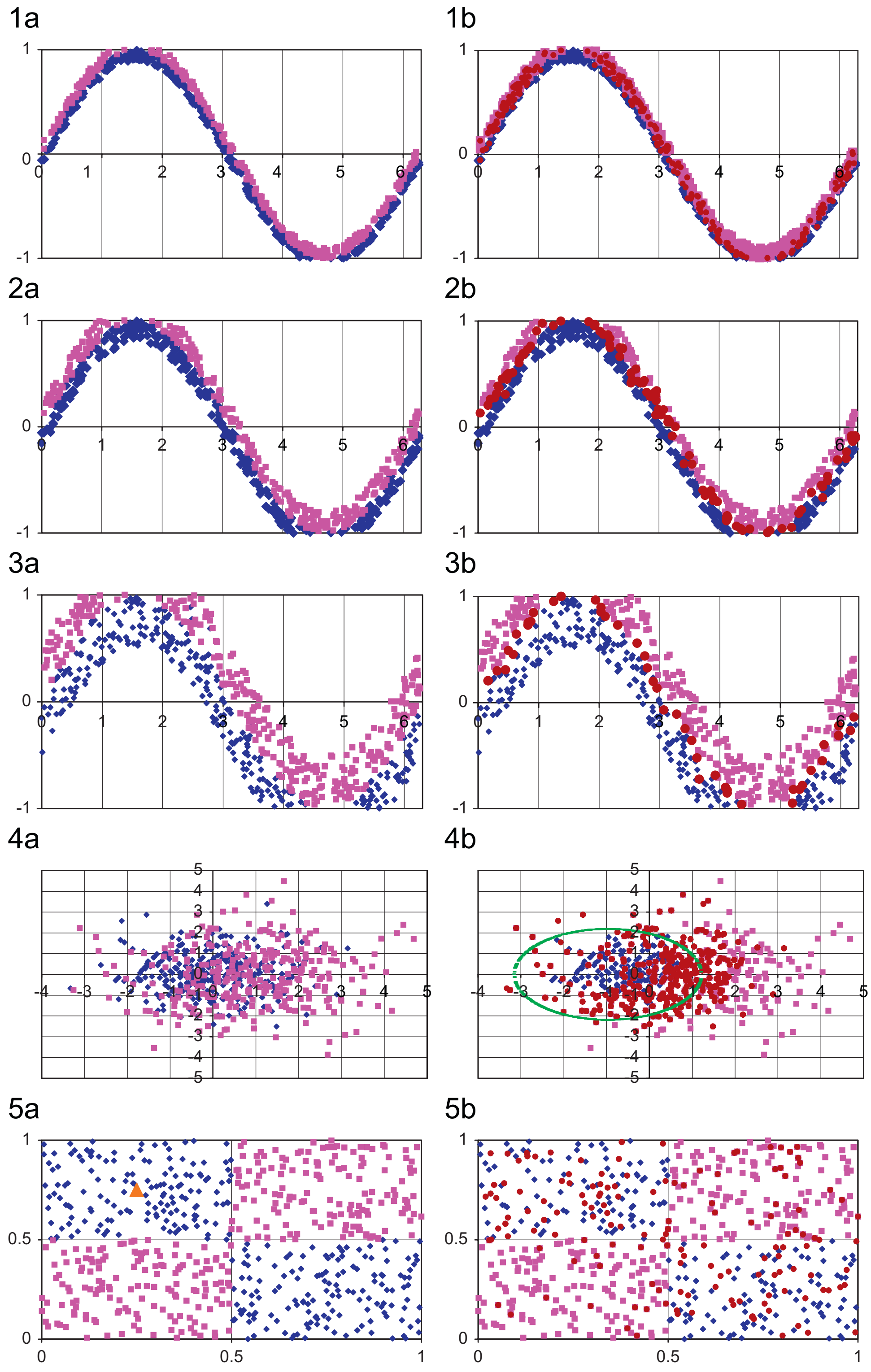

Fig. 3. Five synthetic benchmark data sets. The horizontal and vertical axes denote the 2-dimensional attribute values for each synthetic data set. The left picture (a) shows positive (pink) versus negative (blue) instances in each data set; and the right picture (b) shows instances plus biased points (dark red).

where $X_C = [x_C\ y_C] = \left[\dfrac{\sigma_2^2\mu_{x1} - \sigma_1^2\mu_{x2}}{\sigma_2^2 - \sigma_1^2} \quad \dfrac{\sigma_2^2\mu_{y1} - \sigma_1^2\mu_{y2}}{\sigma_2^2 - \sigma_1^2} \right]$ and $r^2 = \dfrac{\sigma_1^2\sigma_2^2}{\sigma_2^2 - \sigma_1^2} \left[\dfrac{(\mu_{x1} - \mu_{x2})^2 + (\mu_{y1} - \mu_{y2})^2}{\sigma_2^2 - \sigma_1^2} + 4\log(\dfrac{\sigma_2}{\sigma_1}) \right]$.

Given the above conditions, we have $X_C = [-1\ \ 0]^{\mathrm{T}}$ and $r^2 = 4.77$, which offers us the optimal decision boundary (the green circle) in Fig. 3-4(b), with an accuracy of 68.83%.

The optimal decision boundary of the fifth data set, Crs, is exactlya cross which separates uniformly distributed instances into four square areas, with the same-class instances occupying the diagonal squares.

For all five data sets, the biased points (evaluated by using a TB predictor with 10 base classifiers) are marked in dark red. The definition of a biased point is given in Eq. (13).

3.2. The assessment measures

The following measures are used to record LB's performance.

3.2.1. Bias and variance measures

The bias and variance measures have been popularly used in the literature to analyze the behavior of learning algorithms and explain the properties of a classifier ensemble. In short, bias estimates how close the average classifiers produced by a learning algorithm will be to the genuine concept of the underlying data; and variance measures how much the classifiers' predictions will vary from each other. Given a learning set T with N instances and a test set D with M instances, assuming we apply a learning algorithm on a set of bootstrap bags $B_1, B_2, \ldots, B_L$ sampled from T, and build a set of classifiers $C_1, C_2, \ldots, C_L$. Given a test instance x_n whose true class label is y_n, its prediction from a bagging predictor consisting of L base classifiers is denoted by Equation (12). This prediction (y_n^*) is also called a bagging predictor's *main prediction*:

$$y_n^* \leftarrow \underset{y \in Y}{\operatorname{argmax}} \sum_{\ell=1; C_\ell(x_n)=y}^{L} 1. \tag{12}$$

Following the definition in the literature [15,17], we say that an instance (x_n, y_n) is "biased" iff the main prediction y_n^* is different from y_n. The instance is "biased'' in the sense that the learning algorithm has no expertise in identifying it, therefore, its predictions from the majority classifiers are incorrect. Formally, the bias of the instance (x_n, y_n) is denoted by Eq. (13):

$$bi(T, x_n)_{C_1}^{C_L} = \begin{cases} 0 & \text{if } y_n^* = y_n \\ 1 & \text{if } y_n^* \neq y_n. \end{cases} \tag{13}$$

The bias definition in Eq. (13) actually denotes a bagging prediction error, which is assumed to be the minimal error a classifier can possibly approach. Given a sufficiently large L value, it is expected that this bias measure can evaluate the classification "offset'' produced by a learning algorithm with respect to the genuine concept of the underlying data [15,17]. Given a data set, the biased instances produced from different classifiers vary, due to inherent differences among alternative learning algorithms. Therefore, the classifiers' ability varies in handling different types of learning problems, for instance, a biased instance for C4.5 might not be a biased one for kNN at all. Fig. 3 lists all biased points for each synthetic data set by using a 5-fold cross validation. A biased point is evaluated by a TB predictor with $L = 10$ base classifiers (C4.5).

Similarly, the variance of an instance (x_n, y_n) is denoted by Eq. (14), which defines the dispersion of a learning algorithm in making predictions. The smaller the variance value, the more the base classifiers tend to agree on one decision:

$$va(T, x_n)_{C_l}^{C_L} = \frac{1}{L} \sum_{\ell=1:C_\ell(x_n)\neq y_n^*}^{L} 1. \tag{14}$$

Both bias and variance definitions depend on the main vote of the bagging prediction, y_n^*. We therefore introduce an entropy-based *prediction uncertainty* (pu) measure, defined by Eq. (15), where $|Y|$ denotes the total class numbers in T, and p_y is the normalized histogram of L base classifiers' predictions (the percentage of agreeing on class y out of L classifiers). Obviously, the higher the certainty of the classifiers, the lower is the prediction uncertainty value. In the case that all classifiers agree on one particular class, pu reaches the minimal value 0.

$$pu(T, x_n)_{C_1}^{C_L} = -\sum_{y=1}^{|Y|} p_y \log p_y. \tag{15}$$

For a test set D with M instance, we can aggregate the bias, variance, and prediction uncertainty over all M instances, as defined by Eqs. (16)–(18):

$$BI(T, D)_{C_1}^{C_L} = \frac{1}{M} \sum_{n=1}^{M} bi(T, x_n)_{C_1}^{C_L} \tag{16}$$

$$VA(T, D)_{C_1}^{C_L} = \frac{1}{M} \sum_{n=1}^{M} va(T, x_n)_{C_1}^{C_L} \tag{17}$$

$$PU(T, D)_{C_1}^{C_L} = \frac{1}{M} \sum_{i=1}^{M} pu(T, x_n)_{C_1}^{C_L}. \tag{18}$$

Furthermore, to observe the variance and prediction uncertainty of the base classifiers on biased and unbiased instances, respectively, we follow the definitions proposed by Kong and Dietterich [15] and Valentini and Dietterich [17], and calculate biased and unbiased variance and prediction uncertainty values, as defined by Eqs. (19)–(22), where M_b and M_u denote the number of "*biased*'' and "*unbiased*'' instances in the data set D, respectively:

$$VA_b(T, D)_{C_1}^{C_L} = \frac{1}{M_b} \sum_{\{x_n | bi(T, x_n)_{C_1}^{C_L}=1\}} VA(T, x_n)_{C_1}^{C_L} \tag{19}$$

$$VA_u(T, D)_{C_1}^{C_L} = \frac{1}{M_u} \sum_{\{x_n | bi(T, x_n)_{C_1}^{C_L}=0\}} VA(T, x_n)_{C_1}^{C_L} \tag{20}$$

$$PU_b(T, D)_{C_1}^{C_L} = \frac{1}{M_b} \sum_{\{x_n | bi(T, x_n)_{C_1}^{C_L}=1\}} PU(T, x_n)_{C_1}^{C_L} \tag{21}$$

$$PU_u(T, D)_{C_1}^{C_L} = \frac{1}{M_u} \sum_{\{x_n | bi(T, x_n)_{C_1}^{C_L}=0\}} PU(T, x_n)_{C_1}^{C_L}. \tag{22}$$

In practice, bias, variance and prediction uncertainty values are determined by a number of factors, including the training set T, the number of bags L, the sampling mechanisms to construct bags, and the underlying learning algorithm. Because of this, the observations would be more accurate if we can fix some conditions, and compare classifiers based on one attribute dimension only. For this purpose, we fix T, D, L and the learning algorithm in our study, and only compare classifier bias, variance, and prediction uncertainty *w.r.t.* different bootstrap replicate construction approaches (LB vs. TB).

3.2.2. Bag and instance level measures

The disadvantage of the above bias, variance, and prediction uncertainty measures is that they could not explicitly tell whether adding kNN into a bootstrap bag can indeed lead to a better base classifier, with respect to each single bag. We thus introduce three measures at bag and instance levels. The first measure counts the number of times LB wins or loses as a consequence of adding kNN into the bootstrap bag (compared with a same size bag built by TB). Given a data set T with N instances, we first sample $N - K$ instances from T, and denote this subset by P. After that, we sample another

K instances from T, and aggregate these K instances with P to form a TB bag B_i. Meanwhile, we randomly sample K instances from kNN subset S, and aggregate these K instances with P to form an LB bag B'_i. The reason that we use such an approach is to minimize the impact of the random process in bootstrap sampling and thus to clearly see whether LB can improve each bag (by adding kNN) or not. We compare classifiers built by each pair of B_i and B'_i. If the classifier from B'_i(denoted by C'_i) is better than the one from B_i (denoted by C_i), we say LB bag wins. If adding kNN leads to a less accurate learner, we say LB bag loses. The results of this measure are denoted by "win/lose'' in Table 3.

The Q statistic [18] analysis is a tool to assess the consistency among two or more classifiers. Denoting C_i and C_k as two classifiers, their Q statistic is defined by

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}. \tag{23}$$

N^{11} and N^{00} in Eq. (23) denote the number of instances for which C_i and C_k agree on their predictions (both correct or both wrong). N^{10} means the number of instances that C_i predicts correctly but C_k predicts wrongly. The same logic extends to N^{10}. The value of $Q_{i,k}$ varies between -1 and 1. A pair of classifiers that tend to identify the same objects correctly will receive positive Q value. For statistically independent classifiers, their expected Q value is 0. For more than two classifiers, their Q statistic value can be calculated by averaging across all pairs of classifiers.

In our experiments, for any bag B_i (built by traditional bootstrap sampling), we calculate the Q statistic value between the classifiers built by B_i and B'_i respectively (denoted by C_i and C'_i). We then average the Q statistic values over a certain number of bag pairs (B_j and B'_j, $j = 1, 2, \ldots$) and denote the value by *Q-value* in Table 3. For example, *Q-value* (LB vs. TB) means the Q statistic value between the base classifiers built by LB and TB respectively, and *Q*-value (TB vs.TB) means the average Q statistic value between any pair of base classifiers built by TB bags (B_j and B_i, $i \neq j$). We expect that this comparison will help verify the high consistency of the classifiers built by B_i and B'_i, and the relatively low consistency of the classifiers built by any two bags B_i and B_j, $i \neq j$.

To further study classifiers built by B'_i and B_i at the instance level, we first calculate two values: (1) the percentage of instances correctly classifiedby C'_i but incorrectly classified by C_i and (2) the percentage of instances correctly classified by C'_i but incorrectly classified by C_i. We further calculate the difference between these two values and denote the result by ESD: LB vs. TB in Table 3, which stands for *exclusive set difference*. Meanwhile, we randomly select two classifiers C_i and C_j, $i \neq j$, which are built by traditional bootstrap bags, and calculate the differences between (1) the percentage of instances correctly classified by C_i but incorrectly classified by C_j and (2) the percentage of instances correctly classified by C_j but incorrectly classified by C_i. We denoted this value by ESD: TB vs. TB in Table 3.

For each benchmark data set, we will also report its variance, prediction uncertainty, and instance level comparison (ESD) with respect to biased and unbiased instances.

3.3. Why and when LB works

We carry out the following design in all our experiments. Given a data set, we employ 5-fold cross validation. In each fold, for a training set T with N instances, we first sample $N - K$ instances from T, and denote this subset by P. After that, we sample another K instances from T, and aggregate these K instances with P to form the first TB bag B_1. Meanwhile, we randomly sample K instances from the kNN subset S, and aggregate these K instances with P to form the first LB bag B'_1. The reason we are using such an approach is to minimize the impact of the random process in bootstrap sampling so that we can clearly see whether LB can improve each bag by adding kNN.

We build $L = 10$ base classifiers for TB and LB, respectively, and report the average accuracy of TB, LB, and a single unprunned C4.5 decision tree on the test set. Meanwhile, we also report t-test results (p-values) between LB and TB at the 5% critical level to ensure that our observations are statistically significant. The results on accuracy, bias, variance and prediction uncertainty are reported in Table 2. Table 3 reports bag and instance level comparison results. The average number of win/lose bags (over 10 base classifiers) are report in Rows 1 and 2. Row 3 lists the results of the average Q statistic value between classifiers built by B_i and B'_i, and Row 4 represents the results of the average Q statistic value between classifiers built by B_i and any other randomly selected bag B_j ($j = 1, \ldots 10; j \neq i$). Rows 5 to 6 in Table 3 report the comparisons of the exclusive set differences.

In Table 4, variance, prediction uncertainty, and instance level comparisons are calculated from two instance subsets: biased and unbiased instances. Instead of calculating the *VA*, *PU* and *ESD* values over all instances (as reported in Tables 2 and 3), we calculate those values on the collections of biased and unbiased instances, respectively.

3.3.1. Why LB works

The results from Table 2 indicate that LB is statistically significantly better than TB on four data sets ($S_{0.1}$, $S_{0.2}$, $S_{0.5}$ and *Crs*), and is statistically equivalent to TB on Nrm. Among four data sets, LB's absolute accuracy improvement (compared to TB) varies from 0.86% ($S_{0.5}$) to 19.35% (*Crs*). Meanwhile, for either TB or LB, their accuracy improvement (compared to C4.5) is not determined by the accuracy of the base classifier. Neither low nor high accuracy of the base classifiers can guarantee the improvement of a classifier ensemble. This is consistent with observations from Kuncheva and Whitaker [19]:

Table 2
Accuracy, bias, variance and prediction uncertainty comparisons on synthetic data sets (BI: bias, VA: variance, and PU: prediction uncertainty)

Measures		$S_{0.1}$	$S_{0.2}$	$S_{0.5}$	*Nrm*	*Crs*
Accuracy	C4.5	55.13	66.72	90.57	63.25	46.95
	LB	71.87	87.62	92.98	64.95	95.18
	TB	63.85	84.77	92.12	65.03	75.83
	p-Value	<0.001	<0.001	0.006	0.578	<0.001
BI	LB	0.281	0.124	0.070	0.350	0.048
	TB	0.362	0.152	0.079	0.349	0.242
VA	LB	0.245	0.141	0.058	0.144	0.161
	TB	0.283	0.197	0.079	0.152	0.339
PU	LB	0.671	0.364	0.178	0.429	0.517
	TB	0.749	0.523	0.242	0.446	0.871

All p-values less than 0.001 are denoted by <0.001.

Table 3
The bag and instance level comparison on synthetic data sets

Measures		$S_{0.1}$	$S_{0.2}$	$S_{0.5}$	*Nrm*	*Crs*
Win/lose	LB_{win}	8.08	7.60	6.94	4.46	7.58
	LB_{loss}	1.46	2.04	2.12	4.58	1.90
Q-value	LB vs. TB	0.748	0.790	0.915	0.906	0.773
	TB vs. TB	0.245	0.461	0.775	0.750	0.064
ESD	LB vs. TB	0.052	0.053	0.019	0.000	0.214
	TB vs. TB	0.005	0.009	0.002	0.001	0.009

Win/lose: win/lose comparisons of LB bags compared to TB bags. *Q-value*: *Q*-statistic comparison of learners built by LB and TB bags. *ESD*: exclusive set difference which accounts for the differences between (a) the percentage of instances correctly classified by classifier A but incorrectly classified by classifier B and (b) the percentage of instances correctly classified by classifier B but incorrectly classified by classifier A.

Table 4
Variance, prediction uncertainty and exclusive set difference comparisons on synthetic data sets

Measures		$S_{0.1}$	$S_{0.2}$	$S_{0.5}$	*Nrm*	*Crs*
VA	LB: VA_u	0.202	0.115	0.043	0.126	0.120
	TB: VA_u	0.251	0.172	0.062	0.133	0.297
	LB: VA_b	0.319	0.275	0.254	0.178	0.261
	TB: VA_b	0.339	0.331	0.293	0.187	0.394
PU	LB: PU_u	0.577	0.364	0.136	0.382	0.417
	TB: PU_u	0.686	0.523	0.200	0.400	0.809
	LB: PU_b	0.832	0.751	0.701	0.514	0.712
	TB: PU_b	0.859	0.845	0.765	0.531	0.886
ESD	LB vs TBu	0.017	0.039	0.014	−0.011	0.139
	TB vs. TBu	0.005	0.001	0.002	0.000	−0.001
	LB vs. TBb	0.035	0.014	0.005	0.011	0.079
	TB vs. TBb	0.000	0.007	−0.000	0.001	0.010

The results are calculated with respect to biased (subscript b) and unbiased (subscript u) instances, respectively.

low-accuracy base classifiers do not necessarily mean high diversity among the classifiers.

According to definitions in Eqs. (13) and (16), a learner's bias is determined by the error rate of the bagging predictor built by the learner. Therefore, the bias in Rows 5 and 6 of Table 2 are just LB and TB$'s$ error rate, respectively. By putting a small amount of nearest neighbors into the bags built for each test instance, it is clear that LB can reduce base classifier bias. This is further supported by the bag-to-bag comparisons in Rows 1 and 2 in Table 3, where adding kNN most likely leads to a better base classifier except for on the "Nrm" data set, which we will further address in the next subsection. The Q statistic values in Table 3 indicate that the classifier pairs built by bags B_i' and B_i have a high Q statistic value across all data sets, which means that LB only slightly changes the classifier by adding kNN into the traditional bootstrap bags. The instance level comparisons in Rows 5 to 6 in Table 3 further attest that by adding kNN into the bootstrap bags, a classifier built by B_i' will make less mistakes compared with a classifier built by B_i. Take the results of $S_{0.1}$ as an example. The value 0.052 in Table 3 indicates that the classifier C_i built by the traditional bootstrap bag B_i will have 5.2% more chance of making mistakes, compared to a classifier C_i' built by the same size lazy bag B_i'. Given 120 test instances in $S_{0.1}$, this is equivalent to 6.24 or more instances to be misclassified by C_i. Because of the error reduction at the base classifier level, we can observe the accuracy improvement of LB over TB across most benchmark test sets. Therefore, our observations suggest that LB can reduce base classifier bias by simple data manipulation on bootstrap bags.

Now let us further investigate classifier variance and prediction uncertainties. It is clear that LB offers significant reduction of variance and prediction uncertainty in $S_{0.1}$, $S_{0.2}$, $S_{0.5}$ and *Crs*, which are the four data sets where LB outperforms TB. Meanwhile, the amount of variance reduction is likely proportional to LB's accuracy improvement. For example, LB receives the largest variance reduction on Crs, and its accuracy improvement on Crs is also the largest among the four data sets. On the other hand, the variance reduction on *Nrm* is marginal, and the accuracy of LB on Nrm is not better than TB. This observation suggests that adding I_k's neighboring instances into bootstrap bags can reduce base classifiers' variance when classifying I_k, and consequently improve LB's accuracy. The larger the variance reduction, the more the accuracy improvement can possibly be achieved. This observation is consistent with the existing conclusions drawn from the classifier ensembling [10,11], where large variance among base classifiers lead to an inferior classifier ensemble, and reducing base classifier variance is a key to enhance ensemble prediction accuracy.

The variance, prediction uncertainty and instance level comparisons in Table 4, which are based on biased and unbiased instance subsets, indicate that the improvement of LB over TB can be observed on both biased and unbiased instances. Interestingly, when comparing the absolute improvement values over biased and unbiased instances, one can observe a slightly larger improvement on unbiased instances. Take the variance in *Crs* as an example, where the absolute improvement value on biased versus unbiased instances is 0.133 versus 0.177. This suggests that the improvement of LB over TB mainly comes from base classifiers' variance reduction (the enhancement of the prediction certainty) on unbiased instances. This observation not only supports existing conclusions that "variance hurts on unbiased point x but it helps on biased points" [17,20–22], but also verifies that a learner's limitation on some data sets can be improved through the manipulation on the training data. From TB's perspective, unbiased points are those instances on which base classifiers have expertise but occasionally make a mistake. Reducing base classifiers' variance on unbiased instances enhances base classifiers' prediction consistency and therefore improves the classifier ensemble. LB's design ensures that it inherits the bagging property and outperforms TB on unbiased instances. In contrast, since biased points are instances on which base classifiers have no expertise, reducing base classifiers' variance on those instances does not effectively help improve a TB predictor. LB outperforms TB on biased instances by using their kNN to focus on a local region surrounding each biased instance and enforcing a local learning, which may reduce the base classifiers' bias when classifying this instance and generate correct predictions.

In short, the accuracyimprovement of LB mainly results from the fact that carefully trimmed bootstrap bags are able to generate a set of base classifiers with less bias and variance (compared to classifiers trained from pure bootstrap sampled bags). This is, we believe, the main reason why LB works.

3.3.2. When LB works

In this subsection, we argue that (1) from data's perspective, LB mostly works when the majority biased points are surrounded by same-class instances rather than instances from other classes, and (2) from learning's perspective, LB mostly works when adding kNN to the bootstrap bags can lead to better local learning than global learning.

Figs. 3-1(a)–3-3(a) suggest that although $S_{0.1}$, $S_{0.2}$ and $S_{0.5}$ have identical optimal decision surface ($y = \sin(x)$), as training instances move further away from the optimal decision surface, the problem gets easier to solve. This can be confirmed by the accuracies listed in Table 2. Therefore, the data set from $S_{0.1}$, $S_{0.2}$, to $S_{0.5}$ has less biased instances (the biased points are justified by TB). In addition, we can observe that from $S_{0.1}$, $S_{0.2}$, to $S_{0.5}$, the majority biased points are getting closer to the optimal decision surface. This is clearly shown in Fig. 3-3(b) where all 50 biased points of $S_{0.5}$ sit almost right on the optimal decision surface. For a detailed comparison, we enlarge the $\{(0,0),(1,1)\}$ areas in Figs. 3-1(b)–3-3(b) and show the actual points in Figs. 4(a)–(c). The results suggest that from $S_{0.1}$ to $S_{0.5}$, the majority biased points have a decreasing chance of being surrounded by same-class instances. In other words, from $S_{0.1}$ to $S_{0.5}$, a biased instance x_k will have less chance to have neighbors with the same class label as x_k. For example, in Fig. 3-3(c), since all 50 biased points are on the optimal decision surface, on average, the kNN of each biased point x_k will have 50% chance of having a same-class neighbor. When biased points are moving further away from the optimal decision surface, the chance forx_k to have same-class neighbors increases (as shown in Fig. 4 (a) for $S_{0.1}$ data set where many biased points are surrounded by same-class neighbors). An extreme example is shown in the Crs data set (Fig. 5(a)), where almost all nearest neighbors of each biased point have the same class label as

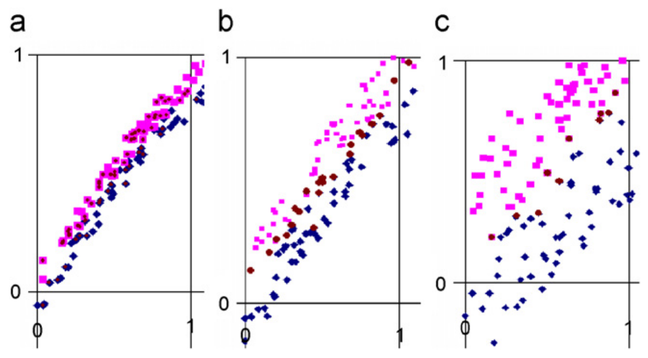

Fig. 4. Enlarged $\{(0,0)(1,1)\}$ rectangle areas for Figs. 3-1(b)–3-3(b). Pink squares and blue diamonds denote positive and negative instances, respectively. Dark red dots denote biased points justified by a bagging predictor (using C4.5). From (a)–(c) it is clear that the majority biased points get closer to the optimal decision surface. As a result, the chance decreases for a biased instance x_k's k nearest neighbors to have the same class label as x_k.

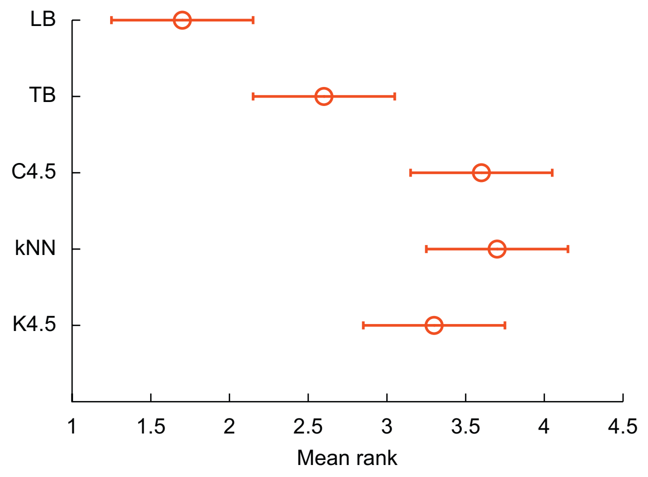

Fig. 5. Applying the Nemenyi test to rival algorithms' mean ranks of reducing classification error. In the graph, the mean rank of each algorithm is indicated by a circle. The horizontal bar across each circle indicates the 'critical difference'. The performance of two algorithms is significantly different if their corresponding mean ranks differ by at least the critical difference. That is, two algorithms are significantly different if their horizontal bars are not overlapping.

the biased instance. Considering that the accuracy improvement of LB over TB gradually decreases from Crs, $S_{0.1}$, $S_{0.2}$, to $S_{0.5}$, we suggest that if the majority biased points are surrounded by same-class neighbors, LB will mostly outperform *TB*.

Now let us turn to the data set Nrm, which consists of two 2-dimensional normal distributions with the green circle in Fig. 3-4(b) denoting the optimal decision boundary. Table 2 shows that a single C4.5 tree achieves an average accuracy of 62.77%, which is reasonably good considering the optimal accuracy (68.83%). TB and LB, however, do not show much improvement. LB actually has worse accuracy than TB, and the p-value shows that LB and TB are statistically equivalent on this particular data set. Thus, for a data set like Nrm, the merits of LB disappear. From Fig. 3-4(b), we can see that the kNN of the majority biased points (210 points) are a mix of both positive and negative instances. The chance for a biased instance to have same-class neighbors is not better than having neighbors from other classes. Therefore, the improvement from LB disappears. This observation complements our previous hypothesis, and we may further infer that LB is likely to be suboptimal when the neighbors of the majority biased points consist of instances from other classes, rather than the class of the biased point themselves.

The above understanding motivates that LB mostly works when adding kNN to the bootstrap bags and can lead to better local learning than global learning. Being 'better' means that, from a test instance's perspective, the learner is able to focus on the small region surrounding the test instance and build a base classifier with a refined decision boundary. If a biased instance's kNN are mainly from the same class, adding them to the bootstrap bags will help a learner build a classifier for a better classification of the biased instance (compared with a globally formed classifier without a local focus). The Crs data set in Fig. 3-5 is a good example to demonstrate that adding kNN improves local learning. The learning task in Crs is essentially an XOR-like problem. A learner like C4.5 has poor capability in solving this type of problem, especially when data are symmetric, because the theme of most tree learning algorithms is to consider a single attribute one at a time and select the best attribute (along with the best splitting point on the attribute) for tree growing. For a data set like Crs, a tree learning algorithm may find no matter which attribute it chooses and wherever it splits the selected attribute, the results are almost the same. As a result, the algorithm may randomly pick up one attribute to split with a random threshold, which consequently generates a classifier with an accuracy close to random guess (50%). TB gains improvement on this data set through bootstrap sampling where randomly sampled instances may form a bag with asymmetric data distributions, which helps a learner to find better splitting than random selection. LB further improves TB by using kNN of each instance to help the learner emphasize the part where one attribute is necessary for best splitting. For example, considering a negative biased instance x_k in the middle of the top left square (the brown triangle) in Fig. 3-5(a), its kNN (about 21 points) all have the same class label as x_k. By adding x_k's nearest neighbors to the bootstrap bag, the learner will thus be able to emphasize the local region and find a good splitting point (closer to (0.5, 0.5)) for tree growing, and the biased instance (originally evaluated by TB) can thus be correctly classified by LB.

The above observations conclude that the niche of LB lies in the conditions that the majority biased points have a large portion of their kNN agree with them, because from learning's perspective, such nearest neighbors can help improve local learning in the emphasized region. This, however, does not necessarily mean that LB's improvement is closely tied to a good kNN classifier, *that is*, LB can outperform TB only on those data sets where kNN classifiers perform superbly. As we have discussed in Section 3.2.1, given one data set, different learning algorithms have different sets of biased instances, depending on how well a learning algorithm can learn the concept in the data set. For LB, as long as the majority biased instances of the base learner have their neighbors largely agree with them, LB can gain improvement over TB. This improvement has nothing to do with whether a kNN can perform well on the whole data set or not. In the following section, we will demonstrate that LB can outperform TB on a large portion of data sets where kNN classifiers perform poorly.

4. Experimental comparisons

Empirical tests, observations, analyses and evaluations are presented in this section.

4.1. Experimental settings

To assess the algorithm performance, we compare LB and several benchmark methods including TB, C4.5, kNN, and K4.5 which is a hybrid lazy learning method combining C4.5 and kNN. We use 10-trial 5-fold cross validation for each data set, employ different methods on the same training set in each fold, and assess their performance, based on the average accuracy over 10 trials. For LB and TB, we use C4.5 unpruned decision trees (Quinlan, 1993) as base classifiers be-

Table 5
Experimental data sets (# of attributes includes the class label)

Data set	# of Classes	# of Attributes	# of Instances
Audiology	24	70	226
Auto-mpg	3	8	398
Balance	3	5	625
Bupa	2	7	345
Credit	2	16	690
Car	4	7	1728
Ecoli	8	8	336
Glass	6	10	214
Hayes	3	5	132
Horse	2	23	368
Ionosphere	2	35	351
Imageseg	7	20	2310
Krvskp	2	37	3196
Labor	4	17	57
Lympho	4	19	148
Monks-1	2	7	432
Monks-2	2	7	432
Monks-3	2	7	432
Pima	2	9	768
Promoters	2	58	569
Sick	2	30	3772
Sonar	2	61	208
Soybean	19	36	683
Splice	3	61	3190
$\text{Statlog}_{\text{Heart}}$	2	14	270
TA	3	6	101
Tictactoe	2	10	958
Tumor	21	18	339
Vehicle	4	19	846
Vote	2	17	435
Vowel	11	14	990
Wine	3	14	178
Wisc. Ca.	2	10	699
Yeast	10	9	1484
Zoo	7	17	101

cause both TB and LB prefer unstable base classifiers. We use 10 base classifiers for TB and LB. LB's K value is determined by fixing the β value to 0.99 for all data sets. Meanwhile, we ensure that at least one nearest neighbor should be added into each bootstrap bag, regardless of the data set size. The way to construct LB bags is exactly the same as what we have described in Section 3.3.

Two single learners (the C4.5 decision tree and kNN) are used to offer a baseline in comparing rival algorithms. The k value of a kNN classifier is exactly the same as LB's K value, which varies depending on the size of the data set. The accuracy of C4.5 is based on the pruned decision tree accuracy.

In addition, the design of employing kNN to customize bootstrap bags lends itself to another lazy learning method with less time complexity (denoted by K4.5 which stands for kNN based C4.5). Given a test instance x_k, find its kNNs from the training set, build a decision tree from its kNNs, and then apply the tree to classify x_k. Intuitively, the decision tree built by x_k's kNNs is superior to a pure kNN classifier in the sense that it can prevent overfitting the local instances surrounding x_k. In our experiments, K4.5 uses the same number of nearest neighbors as LB (which is determined by fixing the β value to 0.99), and the accuracy of K4.5 is based on the pruned C4.5 decision tree.

Our test-bed consists of 35 real-world benchmark data sets from the UCI data repository [23] as listed in Table 5. We use 10-trial 5-fold cross validation for all data sets except for Monks3, on which we use 1/5 instances as training data and 4/5 data as test data because all methods' 5-fold accuracies (except kNN) on Monks3 are 100%.

4.2. Experimental results and statistical tests

Table 6 reports the accuracies of rival algorithms on each benchmark data set, where all data sets are ranked based on LB's *absolute average accuracy improvement* over TB in a descending order. For each data set, the best accuracy achieved among all tested algorithms is bolded, and the second best score is italic. Table 6 also reports the p-values between LB and TB to evaluate whether the mean accuracies of LB and TB are statistically different, and a statistically different value (less than the critical value 0.05) is bolded. Take the first data set *TA* in Table 6 as an example. Because LB has the highest accuracy among all methods, its accuracy value is bolded. The p-value (denoted by <0.001) indicates that the accuracies of LB are statistically significantly higher than that of TB. We then can infer that LB is statistically significantly better than TB on *TA*.

We further proceed to compare rival algorithms across multiple data sets. We deploy the following statistical tests: algorithm correlation, win/lose/tie comparison, and the Friedman test and the Nemenyi test [24].

4.2.1. Algorithm correlation

If we take each method in Table 6 as a random variable, each method's accuracies on all data sets form a set of random variable values. The correlation between any two algorithms can be observed by calculating the correlation between those two random variables. In Table 7, each entry denotes the Pearson correlation between the algorithm of the row compared against the algorithm of the column, which indicates the degree of correlation between the two methods. For example, although both LB and $K4.5$ are based on kNN, the values in the second row indicate that $K4.5$ has much stronger correlation with kNN than LB does.

4.2.2. Win/lose/tie comparison

To statistically compare each pair of algorithms across multiple data sets, a win/lose/tie record is calculated with regard to their classification accuracies as reported in Table 8. This record represents the number of data sets in which one algorithm, respectively, wins, loses to or ties with the other algorithm. A two-tailed binomial sign test can be applied to wins versus losses. If its result is less than the critical level, the wins against losses are statistically significant, supporting the claim that the winner has a systematic (instead of by chance) advantage over the loser.

4.2.3. Friedman test and Nemenyi test

To compare all algorithms in one go, we follow Demsar's proposal [24]. For each data set, we first rank competing algorithms. The one that attains the best classification accuracy is ranked 1, the second best ranked 2, so on and so forth. An algorithm's mean rank is obtained by averaging its ranks across all data sets. Compared with the arithmetic mean of classification accuracy, the mean rank can reduce the susceptibility to outliers that, for instance, allows a classifier's excellent performance on one data set to compensate for its overall bad performance. Next, we use the Friedman test [25] to compare these mean ranks to decide whether to reject the null hypothesis, which states that all the algorithms are equivalent and so their ranks should be equal. Finally, if the Friedman test rejects its null hypothesis, we can proceed with a post hoc test, the Nemenyi multiple comparison test [26], which is applied to these mean ranks and indicates whose performances have statistically significant differences (here we use the 0.10 critical level). According to Table 6, the mean ranks of K4.5 kNN, C4.5, TB and LB are 3.3143, 3.7429, 3.5714, 2.6286, and 1.7429, respectively. Consequently, the null hypothesis of the Friedman test can be rejected, that is, there exists significant difference among the rival algorithms. Furthermore, the Nemenyi test's results can reveal exactly which schemes are significantly different, as

Table 6
Classification accuracy (and standard deviation) comparison of different methods on 35 data sets from UCI data repository (data sets are ranked based on LB's absolute average accuracy improvement over TB in a descending order)

Data set	K4.5 (%)	*k*NN (%)	C4.5 (%)	TB (%)	LB (%)	LB-TB (%)	*p*-Value
TA	47.87 ± 3.21	49.67 ± 4.27	52.05 ± 3.24	*54.29 ± 3.42*	**60.57** ± 2.78	6.28	<**0.001**
Zoo	*93.76 ± 1.78*	93.06 ± 2.41	91.64 ± 1.52	92.38 ± 1.25	**97.09** ± 1.22	4.71	<**0.001**
Sonar	*79.74 ± 2.35*	73.18 ± 6.75	73.08 ± 3.63	75.91 ± 2.21	**80.05** ± 1.78	4.14	**0.001**
Vowel	84.44 ± 1.87	40.51 ± 2.02	79.19 ± 1.57	*86.37 ± 0.99*	**90.19** ± 1.12	3.82	<**0.001**
Autompg	*80.64 ± 1.62*	67.58 ± 1.44	78.08 ± 1.31	77.67 ± 1.72	**81.48** ± 1.46	3.81	<**0.001**
Promoters	*83.02 ± 1.98*	77.36 ± 2.96	74.81 ± 3.85	81.89 ± 2.91	**85.26** ± 1.99	3.37	**0.018**
Tictactoe	91.96 ± 0.79	76.30 ± 1.89	84.53 ± 1.22	*92.09 ± 1.32*	**95.38** ± 0.92	3.29	<**0.001**
Audiology	73.25 ± 1.90	54.42 ± 2.47	76.15 ± 2.04	*78.54 ± 2.71*	**81.75** ± 1.98	3.21	**0.034**
Labor	**88.60** ± 2.27	84.45 ± 2.57	78.95 ± 3.88	84.04 ± 4.25	*86.64 ± 3.51*	2.60	**0.042**
Balance	*82.54 ± 0.69*	**89.28** ± 0.98	65.74 ± 0.85	74.66 ± 0.72	77.23 ± 1.04	2.57	<**0.001**
Hayes	70.38 ± 2.53	64.36 ± 2.68	71.32 ± 3.1	*73.50 ± 2.37*	**75.76** ± 1.18	2.26	**0.005**
Tumor	*41.83 ± 1.02*	**46.18** ± 1.87	40.17 ± 1.90	39.09 ± 1.92	41.21 ± 1.45	2.12	0.340
Lympho	*82.84 ± 2.87*	**83.10** ± 2.38	77.24 ± 2.59	78.67 ± 1.89	80.41 ± 2.02	1.74	**0.031**
Glass	71.92 ± 1.65	64.95 ± 1.87	66.92 ± 2.65	*72.62 ± 1.88*	**74.31** ± 1.62	1.69	**0.022**
Ionosphere	88.69 ± 1.07	84.18 ± 0.89	89.12 ± 1.18	*91.34 ± 0.83*	**92.97** ± 0.74	1.63	**0.014**
Vehicle	72.26 ± 1.29	65.06 ± 1.03	71.56 ± 2.05	*73.41 ± 0.98*	**74.94** ± 1.02	1.53	**0.044**
Wine	95.00 ± 0.84	**96.06** ± 0.93	92.09 ± 1.28	94.47 ± 0.87	*95.62 ± 0.89*	1.15	0.087
Pima	71.85 ± 1.09	75.00 ± 1.05	73.65 ± 1.21	*75.17 ± 0.81*	**76.23** ± 0.68	1.06	0.095
Bupa	61.80 ± 2.94	63.47 ± 2.21	64.20 ± 2.90	*69.17 ± 1.88*	**70.23** ± 2.04	1.06	0.302
Monks3	97.89 ± 0.87	91.49 ± 2.24	97.15 ± 1.21	*98.22 ± 0.69*	**99.24** ± 0.44	1.02	**0.005**
Horse	76.49 ± 1.37	81.61 ± 1.26	**85.22** ± 0.67	84.10 ± 0.94	*84.96 ± 0.63*	0.86	0.360
Monks1	91.85 ± 4.01	87.51 ± 1.74	95.02 ± 3.12	*98.59 ± 0.96*	**99.42** ± 0.72	0.83	**0.029**
ImageSeg.	96.02 ± 0.34	87.24 ± 0.37	96.45 ± 0.24	*96.94 ± 0.47*	**97.74** ± 0.49	0.80	**0.043**
WBreastc	*95.32 ± 0.29*	90.99 ± 0.77	94.51 ± 0.48	94.81 ± 0.60	**95.59** ± 0.51	0.78	0.250
Car	90.03 ± 0.50	78.67 ± 0.64	91.37 ± 0.86	*92.58 ± 0.80*	**93.21** ± 0.48	0.63	0.053
Splice	93.12 ± 0.66	91.66 ± 0.82	93.76 ± 0.56	*94.10 ± 0.43*	**94.64** ± 0.52	0.54	**0.036**
Kr vs. kp	99.19 ± 0.11	89.64 ± 0.15	99.29 ± 0.12	*99.36 ± 0.08*	**99.69** ± 0.08	0.33	**0.026**
Sick	98.52 ± 0.12	94.97 ± 0.14	98.68 ± 0.16	*98.81 ± 0.12*	**99.02** ± 0.13	0.21	**0.046**
Soybean	90.23 ± 0.87	70.26 ± 1.14	90.13 ± 0.77	**91.59** ± 1.01	*91.47 ± 0.87*	−0.12	0.250
Vote	*96.02 ± 0.41*	92.04 ± 0.55	**96.16** ± 0.31	95.91 ± 0.48	95.79 ± 0.33	−0.12	0.270
Credit	84.10 ± 0.97	**86.09** ± 0.95	*85.55 ± 0.98*	85.38 ± 0.88	85.19 ± 0.82	−0.19	0.440
Yeast	54.12 ± 0.93	58.28 ± 0.86	55.59 ± 1.17	**58.84** ± 0.92	*58.47 ± 0.66*	−0.37	0.059
Ecoli	81.43 ± 0.96	**86.01** ± 1.07	82.56 ± 1.17	*83.72 ± 1.06*	83.30 ± 1.11	−0.42	0.279
$\text{Statlog}_{\text{Heart}}$	*79.72 ± 2.14*	**81.48** ± 1.95	75.89 ± 1.58	78.34 ± 2.33	77.77 ± 0.70	−0.57	0.212
Monks2	55.76 ± 2.25	*67.12 ± 0.38*	**67.13** ± 0.92	49.32 ± 2.14	44.74 ± 1.77	−4.58	<**0.001**

For each data set, the best accuracy achieved among all observed methods is bolded, and the second best score is italic. 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.

Table 7
Rival algorithms' Pearson correlation over 35 data sets

Pearson correlation	K4.5	*k*NN	C4.5	TB	LB
K4.5	1	0.777	0.938	0.968	0.958
*k*NN	0.777	1	0.759	0.722	0.681
C4.5	0.938	0.759	1	0.957	0.925
TB	0.968	0.722	0.957	1	0.991
LB	0.958	0.681	0.925	0.991	1

Note that the values are meaningful in a relative way rather than in an absolute way. Such a comparative view indicates whether one method is more correlated with another method considering all alternative methods. (The Spearman rank order correlation also show similar relationships.)

Table 8
Rival algorithms' win/lose/tie records with regard to their classification accuracies across 35 data sets

Win/loose/tie	K4.5	*k*NN	C4.5	TB	LB
K4.5	–				
*k*NN	13/22/0	–			
C4.5	18/17/0	22/13/0	–		
TB	22/13/0	**25/10/0**	**29/6/0**	–	
LB	**28/7/0**	**27/8/0**	**31/4/0**	**28/7/0**	–

Each entry indicates that the algorithm of the row compares against the algorithm of the column. A statistically significant record is indicated in bold.

illustrated in Fig. 5. The mean rank of each scheme is indicated by a circle. The horizontal bar across each circle indicates the critical difference. The performance of two methods is significantly different if their corresponding mean ranks differ by at least the critical difference. That is, two methods are significantly different if their horizontal bars are not overlapping. For example, it is observed that LB is ranked best and is significantly better than alternative methods.

4.3. Experimental result analysis

When comparing K4.5, *k*NN and C4.5 without considering LB and TB, one can easily conclude that they perform comparably and follow a ranking order: K4.5 (3.31), C4.5 (3.57), and *k*NN (3.74). Because of the heavy overlapping of their horizontal bars in the Nemenyi test (Fig. 5), the differences between these methods are not statistically significant, although K4.5 performs relatively better than C4.5 and *k*NN (overall K4.5 wins *k*NN and C4.5 on 22 and 17 data sets, respectively). This asserts that in practice, a lazy learner can indeed outperform eager learners (like C4.5), even though the former is based on a pure *k*NN classifier and the model was formed from a small portion of the training data (notice that K4.5 only forms a decision tree from a small number of instances surrounding a test instance).

The results in Table 6, along with the statistical tests presented in Table 7, Table 8 and Fig. 5 indicate that overall, LB receives the best performance among all five methods. Compared to three single learner methods K4.5, *k*NN and C4.5, LB wins in 28, 27 and 31 data sets, respectively. The Nemenyi test in Fig. 5 also demonstrates that the rank of LB is much better than that of K4.5, *k*NN and C4.5. These

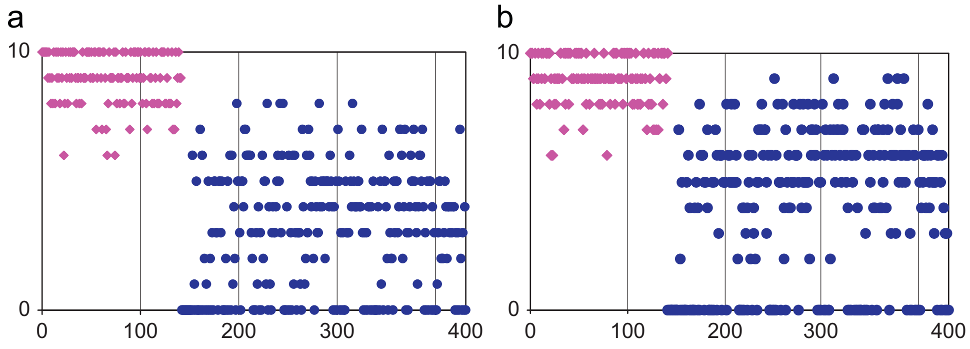

Fig. 6. Number of times each instance is misclassified by TB and LB, respectively (10-trial 5-fold cross validation). The horizontal axis denotes the instance ID and the vertical axis shows the misclassification frequency: (a) TB results on Monks2 (b) LB results on Monks2

results support the claim that LB is statistically significantly better than these three methods. Meanwhile, the win/lose/tie records show that TB can also outperform K4.5, *k*NN and C4.5 more often than not. This suggests that an ensemble learning model is indeed more accurate than a single learner, regardless of whether the base classifiers are trained by eager learning or lazy learning paradigm.

The results in Table 6 indicate that LB can outperform TB on 12 data sets with more than 2% absolute accuracy improvement, and the largest improvement is 6.28% in *TA* (the teaching assistant data set). Overall, LB outperforms TB on 28 data sets, of which the results on 21 data sets are statistically significant. On the other hand, TB outperforms LB on 7 data sets, of which only 1 data set (Monks2) is statistically significant. LB's wins (28) versus losses (7) compared with TB is also statistically significant. In addition, the Nemenyi test results in Fig. 5 also agree that LB is statistically significantly better than TB.

Because LB relies on the *k*NN selected for each test instance to customize bootstrap bags, we must further investigate the relationship between LB's performance and the quality of the selected *k*NN, and clearly answer whether LB's performance crucially relies on a superb *k*NN classifier. This part of the study is carried out based on the algorithm correlation and the win/lose/tie comparison in Tables 7 and 8 respectively. The second row inTable 7 indicates that although LB was built based on *k*NN, it actually has very little correlation with *k*NN. We believe that this is mainly because LB is essentially an ensemble learning framework. Even if the *k*NN were poorly selected, the ensemble nature will essentially smooth the impact and reduce the correlation between LB and *k*NN (one can further investigate the last row in Table 7 and confirm that LB has the highest correlation with TB, mainly because of their ensemble nature). At individual data set level and from the perspective of the win/lose/tie comparison, we would also like to study whether a superior *k*NN classifier can help LB outperform TB, or vice versa; in other words, whether LB can outperform TB more often in data sets where *k*NN wins C4.5/TB than in data sets where *k*NN loses to C4.5/TB. The results in Table 6 indicate that out of the 22 data sets where *k*NN loses to C4.5, LB wins TB in 19 data sets (a probability of $19/22 = 0.864$). On the other hand, out of the 13 data sets where *k*NN wins C4.5, LB wins TB in 9 data sets (a probability of $9/13 = 0.692$). In addition, out of the 25 data sets where *k*NN loses to TB, LB wins TB in 22 data sets (a probability of $22/25 = 0.88$), and out of the 10 data sets where *k*NN wins TB, LB wins TB in 6 data sets (a probability of $6/10 = 0.6$). Considering that all these values (0.864, 0.692, 0.88 and 0.6) are fairly close to each other, we argue that there is no clear correlation between *k*NN and LB for any particular data set. Either a superior or an inferior *k*NN classifier may help LB outperform TB.

Now, let us turn to Monks2 on which LB is worse than TB. There are 432 (142 positive and 290 negative) instances, each of which has 6 multivariate attributes and one class label. The genuine concept is: *Class* = *Positive* iff exactly two attributes have value 1; *otherwise*, *Class* = *Negative*. Obviously, this is an XOR-like problem except that each attribute is multi-valued instead of binary. Both *k*NN and top-down-induction-of-decision-tree type learners, unfortunately, cannot effectively solve this type of problem (as we have explained in Section 3.3.2). *k*NN and C4.5 solve this problem by simply classifying all instances as the negative class, which is equivalent to an accuracy of $290/432 = 67.13\%$ (the results in Table 6 confirm that *k*NN and C4.5's accuracies are very close to this value). According to the definition given in Eq. (13), all positive instances are biased instances. To explain why LB fails in this data set, let us consider biased and unbiased instances separately. Assume for any instance x_k in the data set, its nearest neighbors have only one attribute value different from x_k. Following this assumption and Monks2's genuine concept, one can imagine that for any positive instance (biased instances), over 60%[2] of its *k*NN are from the opposite (negative) class. According to our reasoning in Section 3.3.2, LB is ineffective because adding *k*NN instances to each bag is not helpful in improving local learning. For unbiased instances, because C4.5 essentially cannot solve this type of problem, adding *k*NN will only make the matter worse.

To verify the above understanding, we record the number of times each instance is misclassified in the 10-trial 5-fold cross validation by using TB and LB respectively. We report the histogram of all 432 instances in Fig. 6 where the horizontal axis denotes the instance ID and the vertical axis represents the misclassification frequency. Because we run each algorithm to classify each instance 10 times, the maximal number of times each instance can be misclassified is 10. To make the results straightforward, we have all the 142 positive instances listed from ID 0 to 141, and all the 290 negative instances listed from ID 142 to 431.

As shown in Fig. 6, where (a) and (b) represent the results from TB and LB, respectively, almost all positive instances (pink) are misclassified by both TB and LB, and about 129 (30%) negative instances (blue) were never misclassified by either TB or LB. Because C4.5 is incapable of capturing the underlying concept for Monks2, trimming bootstrap bags provides no help in improving local learning for positive instances. It also makes the classification worse for negative (unbiased) instances since if the learner is incapable of learning the concept, the decision to classify all instances as negative is already optimal. Comparing Figs. 6(a) and (b), we can find a clear increase of the number of times the negative instances are misclassified, which is the consequence of the newly added *k*NN.

[2] This value is calculated by a simplified assumption that each attribute has 3 attribute values {1, 2, 3} and an instance's nearest neighbors only have one attribute value different from the instance.

Notice that the issue raised by Monks2 is extreme in the sense that the learner is incapable of capturing the underlying concept. For data sets where learners may experience difficulties but would not completely fail, the advantage of LB over TB is obvious.

5. Conclusions and future work

In this paper, we have proposed a generic lazy bagging (LB) design, which customizes bootstrap bags according to each test instance. The strength of LB stems from its capability to reduce its base learners' classification bias and variance for each specific test instance. As a result, the bagging predictor built by LB can achieve better performance than the traditional bagging approach (TB). Because LB crucially relies on the selection of k nearest neighbors (kNN) for each test instance, we have further proposed a sampling-entropy-based approach that can automatically determine the k value for each data set, according to a user specified β value that is independent of data sets. Following the sampling entropy, LB trims each bootstrap bag by using k nearest neighbors and ensures that all bags can be as independent as possible (*w.r.t.* the β value). Meanwhile, LB endeavors to take care of each test instance's speciality. Our empirical study on carefully designed synthetic data sets has revealed why LB can be effective and when one may expect LB to outperform TB. Our experimental results on 35 real-world benchmark data sets have demonstrated that LB is statistically significantly better than alternative algorithms K4.5, kNN, C4.5 and TB in terms of minimizing classification error. We have also observed that LB is less powerful only when its base learners are incapable of capturing the underlying concepts.

Because of their lazy nature, lazy learning algorithms have a sub-optimal feature: low classification efficiency. Our future work will focus on how to speed up LB. One direction is to employ incremental classifiers as base learners, for example, na¨ve-Bayes (NB) classifiers. It is trivial to map an existing NB and a new training instance to a new NB that is identical to the NB that would have been learned from the original data augmented by the new instance. In the case of LB, one can train an NB from each original bag during the training time. Upon receiving a test instance and adding its kNN into each bag, one can simply update each NB by these k new training instances. It will be much more efficient than retraining a classifier from scratch in order to incorporate these neighbors. Another direction is to adopt an anytime classification scheme [27,28]. For example, one may specify an order for bag processing. To classify a test instance, one can follow this order of bags to train each classifier in sequence and obtain its prediction until classification time runs out. This is an anytime algorithm in the sense that the classification can stop anywhere in the ordered sequence of bags. In this way, LB can adapt to available classification time resources and obtain classification as accurate as time allows.

References

[1] J. Friedman, R. Kohavi, Y. Yun, Lazy decision trees, in: Proceedings of the AAAI, MIT Press, Cambridge, MA, 1996, pp. 717–724.
[2] D.W. Aha, Editorial: lazy learning, Artif. Intell. Rev. 11 (1–5) (1997).
[3] D. Aha, D. Kibler, M. Albert, Instance-based learning algorithms, Mach. learn. 6 (1991) 37–66.
[4] X.Z. Fern, C.E. Brodley, Boosting lazy decision trees, In: Proceedings of the 20th ICML Conference, 2003.
[5] S.D. Bay, Combining nearest neighbor classifiers through multiple attribute subsets, In: Proceedings of the 15th ICML Conference, 1998.
[6] L. Breiman, Bagging predictors. Mach. Learn. 24 (2) (1996) 123–140.
[7] Y. Freund, R. Schapire, A decision-theoretic generalization of on-line learning and an application to Boosting, Comput. Syst. Sci. 55 (1) (1997) 119–139.
[8] J. Kittler, M. Hatef, R. Duin, J. Matas, On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (3) (1998) 226–239.
[9] C. Mesterharm, Using linear-threshold algorithms to combine multi-class sub-experts, in: Proceedings of the 20th ICML Conference, 2003, pp. 544–551.
[10] L. Beriman, Bias, variance, and arching classifiers, Technical Report 460, UC-Berkeley, CA, 1996.
[11] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting and variants, Mach. Learn. 36 (1999) 105–142.
[12] R. Khoussainov, A. Heb, N. Kushmerick, Ensembles of biased classifiers, in: Proceedings of the 22nd ICML Conference, 2005.
[13] I. Kononenko, Estimating attributes: analysis and extension of RELIEF, in: Proceedings of the ECML Conference, 1994.
[14] G. James, Variance and bias for general loss function, Mach. Learn. 51 (2) (2003) 115–135.
[15] E.B. Kong, T.G. Dietterich, Error-correcting output coding corrects bias and variance, in: Proceedings of the 12th ICML Conference, 1995, pp. 313–321.
[16] R. Kohavi, D. Wolpert, Bias plus variance decomposition for zero-one loss functions, in: Proceedings of the Thirteenth ICML Conference, 1996.
[17] G. Valentini, T.G. Dietterich, Low bias bagged support vector machines, in: Proceeding of the 20th ICML Conference, 2003.
[18] GU. Yule, On the association of attributes in statistics, Philos. Trans. Ser. A 194 (1900) 257–319.
[19] L. Kuncheva, C. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Mach. Learn. 51 (2) (2003) 181–207.
[20] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[21] P. Domingos, A unified bias-variance decomposition, and its applications, In: Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, CA, 2000, Morgan Kaufmann, Los Altos, CA, pp. 231–238.
[22] P. Domingos, A unified bias-variance decomposition for zero-one and squared loss. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence, Austin, TX, 2000, AAAI Press, pp. 564–569.
[23] C.L. Blake, C.J. Merz, UCI Repository of Machine Learning Databases, 1998.
[24] P. Domingos, A unified bias-variance decomposition, Technical Report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2000.
[25] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1) (1940) 86–92.
[26] W. Kohler, G. Schachtel, P. Voleske, Biostatistik, second ed., Springer, Berlin, 1996.
[27] J. Grass, S. Zilberstein, Anytime algorithm development tools, SIGART Artificial Intelligence. vol. 7, no. 2, ACM Press: New York, 1996.
[28] Y. Yang, G. Webb, K. Korb, K.M. Ting, Classifying under computational resource constraints: anytime classification using probabilistic estimators, Machine Learning 69 (1) (2007) 35–53.
[29] X. Zhu, Lazy bagging for classifying imbalanced data, in: Proceedings of the ICDM Conference, 2007.

About the Author—XINGQUAN ZHU is an assistant professor in the Department of Computer Science and Engineering at Florida Atlantic University, Boca Raton, Florida (USA). He received his Ph.D. in computer science (2001) from Fudan University, Shanghai (China). He was a postdoctoral associate in the Department of Computer Science, Purdue University, West Lafayette, Indiana (USA) (February 2001–October 2002). He was a research assistant professor in the Department of Computer Science, University of Vermont, Burlington, Vermont (USA) (October 2002–July 2006). His research interests include data mining, machine learning, data quality, multimedia systems and information retrieval.

About the Author—YING YANG received her Ph.D. in Computer Science from Monash University, Australia, in Year 2003. Following academic appointments at University of Vermont, USA, she currently holds a Research Fellow at Monash University, Australia. Dr. Yang is recognized for contributions in the fields of machine learning and data-mining. She has published many scientific papers and book chapters on adaptive learning, proactive mining, noise cleansing and discretization.