Generalization Performance of Pure Accuracy and its Application in Selective Ensemble Learning

Jieting Wang, *Member, IEEE*, Yuhua Qian, *Member, IEEE*, Feijiang Li, *Member, IEEE*, Jiye Liang, *Senior Member, IEEE*, and Qingfu Zhang, *Fellow, IEEE*

Abstract—The pure accuracy measure is used to eliminate random consistency from the accuracy measure. Biases to both majority and minority classes in the pure accuracy are lower than that in the accuracy measure. In this paper, we demonstrate that compared with the accuracy measure and F-measure, the pure accuracy measure is class distribution insensitive and discriminative for good classifiers. The advantages make the pure accuracy measure suitable for traditional classification. Further, we mainly focus on two points: exploring a tighter generalization bound on pure accuracy based learning paradigm and designing a learning algorithm based on the pure accuracy measure. Particularly, with the self-bounding property, we build an algorithm-independent generalization bound on the pure accuracy measure, which is tighter than the existing bound of an order $O(1/\sqrt{N})$ (N is the number of instances). The proposed bound is free from making a smoothness or convex assumption on the hypothesis functions. In addition, we design a learning algorithm optimizing the pure accuracy measure and use it in the selective ensemble learning setting. The experiments on sixteen benchmark data sets and four image data sets demonstrate that the proposed method statistically performs better than the other eight representative benchmark algorithms.

Index Terms—Generalization performance bound, linear-fractional measure, pure accuracy, selective ensemble learning

1 Introduction

During the process of decision-making, decision makers often make random guesses, which may generate consistency with the true state. We call this kind of consistency as random consistency. Designed by humans and induced from some limited data, the learning algorithms analogously have probabilities to generate randomness [1] and random consistency [2], [3]. However, in the area of classification, simple consistency measures, which are defined as general functions of the entries in confusion matrix, are generally used to evaluate the performance of the algorithms. For instance, the accuracy measure is the summation of the true positive and the true negative, the F-measure and the G-mean measure are defined as the harmonic mean and the geometrical mean of the precision rate and the recall rate, respectively. They do not take the random consistency into consideration. Evaluation results with containing the random consistency may lead to a deceptive feedback loop and then make an influence on the improvement of the learning system.

Nowadays, eliminating random consistency from consistency measures has become a hot research topic. This topic originated from the area of educational psychology [4], [5], [6], [7]. The researchers anticipated the expected score of the accurate answer not by reason and logic would be zero instead of one. With the zero baseline for a right guess, reviewer may improve the reliability of assessment and facilitate the examinees to increase their performance in a proper way. Various approaches to correct chance agreement have been proposed. One of the most used ones is to penalize the right scores by wrongs with a factor that can ensure the expected score of a pure guess is zero. Another one is to define statistics that penalize the total agreement by the associativity degree of examinees' marginal distributions, such as Cohen's κ [8], Scott's π [9], Goodman and Kruskal's λ [10], Mak's ρ and Hamann's η [11]. These statistics are built for hypothesis testing in inferential statistics and used to make inferences about some unknown distribution parameters.

In the field of clustering, eliminating random consistency recently serves as an important technique to evaluate the quality of clustering results. Adjusted Rand Index, a popular evaluation measure in clustering, is defined by eliminating random

- *Jieting Wang and Feijiang Li are with the Institute of Big Data Science and Industry, Shanxi University, Taiyuan, Shanxi 030006, China. E-mail: {jietingwang, feijiangli}@email.sxu.edu.cn.*
- *Yuhua Qian is with the Institute of Big Data Science and Industry, Shanxi University, Taiyuan, Shanxi 030006, China, and also with the Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Shanxi University, Taiyuan, Shanxi 030006, China. E-mail: jinchengqyh@126.com.*
- *Jiye Liang is with the Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Shanxi University, Taiyuan, Shanxi 030006, China. E-mail: ljy@sxu.edu.cn.*
- *Qingfu Zhang is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China. E-mail: qingfu.zhang@cityu.edu.hk.*

Manuscript received 12 Dec. 2019; revised 4 Mar. 2022; accepted 22 Apr. 2022. Date of publication 29 Apr. 2022; date of current version 6 Jan. 2023.
This work was supported in part by the Key Program of National Natural Science Foundation of China under Grant 62136005, in part by the National Key Research and Development Program of China under Grant 2021ZD0112402, in part by the National Natural Science Foundation of China under Grant 62106132, and in part by the Program for the San Jin Young Scholars of Shanxi.
(Corresponding author: Yuhua Qian.)
Recommended for acceptance by S. Zilles.
Digital Object Identifier no. 10.1109/TPAMI.2022.3171436

consistency from the Rand Index [12]. Some other similarity measures like Jaccard coefficient and information-entropy based measures are also promoted as correction measures [13], [14], [15]. Furthermore, the information-entropy based measures after correction show unbiasedness to the number and the size of clusters [13], [14], [15]. As far as we know, the above mentioned measures are often used in the final evaluation stage and do not participate in the process of learning.

A learning problem is a problem that considers two topics: designing algorithms that can automatically improve a performance measure from the experience data and estimating the generalization ability of the learning algorithms. In previous decades, the accuracy measure has been the fundamental performance measure in learning. It has been well-studied that the traditional algorithms, including logistic regression, support vector machine and Adaboost are designed to optimize convex surrogate loss functions of the error probability (one minus accuracy) [16], [17]. And in ensemble learning, accuracy has been used as the preferential measure to evaluate the performance of integration [18], [19]. In addition, almost all learning theories focus on searching the generalization bounds with respect to the error probability [20], [21], [22], [23], [24].

With the interest in eliminating random accuracy from the accuracy measure, Wang et.al [2] has defined a pure accuracy measure (PA). Some classifiers that optimize PA have been developed, which contain plug-in rule [2] and support vector machine model (SVM) [3]. In [2], it has been proved that learning by PA implies a good A by showing the lower bound and the upper bound of the error probability of the optimal rule in the sense of PA. In addition, the learnability of PA has been shown in both finite and infinite hypothesis spaces.

However, the proposed plug-in rule is incapable of providing a group of weight to show the importance of input features, which hinders its use feature selection or in selective ensemble learning. The proposed SVM model approximately maximizes PA with fixing the positive probability of the output label larger than that of the training label. In this paper, we attempt to design an algorithm that optimizes PA, which applies to the learning settings with a linear combination of input as the decision function. We also aim to survey the generalization performance of PA by a tighter generalization bound.

The pure accuracy measure belongs to the class of linear-fractional performance measures. Such measures include the well-known Jaccard coefficient, F-measure and so on. The linear-fractional measures are non-decomposable, namely, they cannot be represented as a summation over individual instances [25], [26], [27]. This property causes difficulties in estimating the measures without bias and the failure of the traditional gradient methods in optimizing them.

Recently, much interest has been put in the linear-fractional measures, including some theoretical analysis on the surrogate consistency between the linear-fractional measures with other losses [25], [26], [28], the consistency analysis of the two-step approach to optimize the linear-fractional measures [29], [30], [31], and the generalization performance analysis of the linear-fractional measures [32]. It is worth mentioning that Dembczyński et.al [32] provided a Rademacher complexity based generalization bound with an order of $O(1/\sqrt{N})$ for the linear-fractional measures by applying the property of p-Lipsitz continuity and the Hoeffding's inequality.

In designing algorithms, there are two types of methods: indirect method and direct method. Indirect methods proceed by formalizing a sequence of cost-sensitive classifiers or plug-in rules to optimize the linear-fractional measures [29], [30], [31], [33], [34], [35]. But the cost-sensitive classifiers are trapped in the determination of combination coefficient for losses from different classes, and the plug-in rules need more training data because it is a two-step process consisting of learning the posterior probability and searching the truncated threshold to maximize the specified measure. Direct methods directly connect the model parameters with the measures [27], [36], [37]. AMP (Alternate Maximization Procedure) method [36] used the lower level sets of the the linear-fractional measures and alternatively maximized the level value and the model parameters to optimize the measures. Bisection method [31] used the same idea as AMP method and searched the optimal level value in binary. These two methods used the same strategy that introduced a parameter to transform the problem of optimizing the interesting measure to a linear one. This strategy is efficient but may fail when the objective function also contains a regularization term. SVMperf [37] optimized a convex relaxation of the interested measure, which required that the margin between the true label vector and the other possible label vectors should be larger than the score evaluated by the specified measure. Gradient method [38], [39] is an extension of SVMperf. It inferred two label vectors to define the gradient of the linear-fractional measure and searched the optimal model parameter by gradient descent method. SVMperf and Gradient are novel but are restricted by the large computational complexity. Although the above algorithms effectively optimize the linear-fractional measures, there still exists much room to promote the optimization performance.

From the above reviews, we aim to develop a tighter generalization bound and design a more direct learning algorithm on the pure accuracy measure. The major contributions of this paper are summarized as follows:

- First, we build a generalization bound on PA based on the Rademacher complexity and the number of instances N. It declines faster than $O(1/\sqrt{N})$ with considering the self-bounding property.
- Second, we design a method that directly optimizes the pure accuracy measure. This method is used in the scene of ensemble learning, and the corresponding algorithm is called PASE. PASE does not use a relaxation loss and can be extended to optimize the models containing regularization terms. Experimental results demonstrate the effectiveness of PASE.

The organization of this paper is as follows. We give the definition and the advantages of pure accuracy in Section 2. In Section 3, an existing bound on pure accuracy is reviewed, then a tighter concentration inequality and a tighter generalization bound on PA are shown. In Section 4, first, the benchmark algorithms are reviewed. Then PASE is presented and its performance is validated through sixteen benchmark data sets and four image data sets. Section 5 concludes this paper. All the proofs and some experimental results are provided in the Supplementary Material, which can be found on the Computer

TABLE 1
Confusion Matrix

$h(X)$ \ Y	$Y=+1$	$Y=-1$	Total(h)
$h(X)=+1$	TP	FP	$q(h)$
$h(X)=-1$	FN	TN	$1-q(h)$
Total(Y)	p	$1-p$	1

Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2022.3171436.

2 Pure Accuracy and Its Advantages Over Accuracy

In this paper, binary classification is considered. It aims to learn a classifier $h(X)$ mapping from the feature space $X \in \mathcal{X} \subseteq \mathcal{R}^d$ to the binary label space $Y \in \mathcal{Y} = \{+1, -1\}$. The classifiers are learnt from a hypothesis space $\mathcal{H}$. To evaluate the classifiers, the confusion matrix is often used, which is defined as Table 1.

In Table 1

$$TP = \mathbb{P}_{X,Y}(h(X) = +1, Y = +1), \tag{1}$$
$$FP = \mathbb{P}_{X,Y}(h(X) = +1, Y = -1), \tag{2}$$
$$FN = \mathbb{P}_{X,Y}(h(X) = -1, Y = +1), \tag{3}$$
$$TN = \mathbb{P}_{X,Y}(h(X) = -1, Y = -1), \tag{4}$$

and the positive class probabilities of h and Y are

$$q(h) = \mathbb{P}_X(h(X) = +1), p = \mathbb{P}_Y(Y = +1), \tag{5}$$

respectively. In a learning task, p is taken as a constant, and $q(h)$ is an unknown quantity w.r.t the output of the classifier.

Based on the confusion matrix, the accuracy measure (A) and the error probability (L) are defined as

$$A(h(X), Y) = \mathbb{P}_{X,Y}(h(X) = Y) = TP + TN, \tag{6}$$
$$L(h(X), Y) = \mathbb{P}_{X,Y}(h(X) \neq Y) = FP + FN, \tag{7}$$

respectively.

The underlying probability distribution of $\mathcal{X} \times \mathcal{Y}$ is usually unknown. We only have a collection of empirical data drawn independently from it, denoted as $\mathcal{S}_N = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$. In the empirical situation, the letter with a hat and a subscript N denotes the corresponding empirical measure. For example, the empirical measure of TP is

$$\widehat{TP}_N = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[h(\boldsymbol{x}_i) = +1, y_i = +1], \tag{8}$$

where $\mathbb{I}$ is the indictor function.

2.1 A Framework of Pure Consistency Measure

To evaluate the performance of the classifier, a consistency measure (CM) is usually introduced.

Definition 1. *For two random discrete variables Z_1 and Z_2, a measure is a CM, if for $\forall\ z \in \mathcal{Z}$, it increases monotonically with $p_z = \mathbb{P}_{Z_1,Z_2}(Z_1 = z, Z_2 = z)$, where $\mathcal{Z}$ is the domain of the variables.*

The accuracy increases monotonically with TP and TN. Then, A is a consistency measure of $h(X)$ and Y, while the error probability is not.

For two discrete variables, if at least one of them is a totally random variable, there also exists consistency between them. Consistency generated by randomness rather than by logic may influence the objectivity and reliability of the evaluation result. To measure this kind of consistency, random consistency measure (RCM) is defined as follows.

Definition 2. *A measure is a RCM if it can measure the consistency that is generated by randomness.*

To measure the pure consistency, a general framework of pure consistency measure (PCM) is defined as follows [8], [12].

Definition 3. *For two random discrete variables Z_1 and Z_2, PCM is defined as*

$$PCM(Z_1, Z_2) = \frac{CM(Z_1, Z_2) - RCM(Z_1, Z_2)}{\max_{Z_1,Z_2} CM(Z_1, Z_2) - RCM(Z_1, Z_2)}. \tag{9}$$

The framework of PCM is to subtract a random consistency measure from the original consistency measure and then to normalize the maximal value to 1 by an upper bound. When PCM=0, the consistency of Z_1 and Z_2 is considered to be generated completely by randomness.

2.2 The Definition of Pure Accuracy

We give the concrete formulation of RCM and PCM in the context of accuracy, and call the random consistency in accuracy as Random Accuracy (RA) and the eliminated measure as Pure Accuracy (PA). How to define RA is crucial to formulize PA.

Here, we follow the definition of RA in [2], [3], where RA is defined as the mean accuracy over a binary partition class. For the classifier $h(X)$ to be evaluated, the binary partition class $\mathcal{H}_{q(h)}$ contains all the possible binary partitions that have the same class distribution as $h(X)$. That is

$$\mathcal{H}_{q(h)} = \{h' | \mathbb{P}_X(h'(X) = +1) = q(h), h'(X) \in \{+1, -1\}\}. \tag{10}$$

Different with partitions in [40], the partitions in $\mathcal{H}_{q(h)}$ are independent of X. For N instances, a partition in $\mathcal{H}_{q(h)}$ is a 1 by N vector with each element taking values from $\{+1, -1\}$. Each element corresponds to a predict label of each instance.

In [2], it has been proved that when the partitions h' in $\mathcal{H}_{q(h)}$ follow the uniform distribution, their true positive number $N \cdot \widehat{TP}_N(h')$ follows the hypergeometric distribution with the parameter $Nq(h)$ and Np. That is

$$N \cdot \widehat{TP}_N(h') \sim \mathbb{P}_{HG}(Nq(h), N, Np), \tag{11}$$

where the function $\mathbb{P}_{HG}(Nq(h), N, Np)$ represents the probability of obtaining exactly $N \cdot TP(h')$ positive instances and $Nq(h) - N \cdot TP(h')$ negative instances if $Nq(h)$ instances are chosen at random without replacement from a finite

population containing N instances of which Np are the number of positive instances and $N - Np$ are the number of negative instances. Thus, according to the properties of hypergeometric distribution, the expectation value[1] of $\widehat{TP}_N(h')$ is

$$\mathbb{E}_{h':h'\in\mathcal{H}_{q(h)}}\widehat{TP}_N(h') = pq(h). \tag{12}$$

The true negative number $N \cdot TN(h')$ follows the hypergeometric distribution with the parameters $N(1-q(h))$ and $N(1-p)$

$$N \cdot \widehat{TN}_N(h') \sim \mathbb{P}_{HG}(N(1-q(h)), N, N(1-p)), \tag{13}$$

and

$$\mathbb{E}_{h':h'\in\mathcal{H}_{q(h)}}\widehat{TN}_N(h') = (1-p)(1-q(h)). \tag{14}$$

Thanks to the partitions having the same class distribution and following the uniform distribution, the expectation of their $\widehat{TP}_N(h')$ and $\widehat{TN}_N(h')$ have the above simple formulations. Then the random accuracy has the following simple formulation.

Definition 4 ([2], [3]). *RA of $h(X)$ is defined as*

$$RA(h) = pq(h) + (1-p)(1-q(h)). \tag{15}$$

Definition 5 ([2], [3]). *Under the framework of PCM, PA of $h(X)$ is defined as*

$$PA(h(X), Y) = \frac{A(h(X), Y) - RA(h(X), Y)}{1 - RA(h(X), Y)}. \tag{16}$$

Note that the formulation of PA coincides with the definition of Cohen's κ statistic [8], which measures the chance agreement between two raters by assuming the raters are statistically independent. However, the way we define RA may inspire the other new formulations by attaching more complex distribution on $\mathcal{H}_{q(h)}$ or using the other estimators except expectation over the partitions.

In the area of detecting the dependence between functions, the indeed independence statistics are derived, which satisfy the property that their value are zero when the functions are statistically independent. Based on the statistics, superior independence test and regression method have been developed [41], [42]. The RA measure is a kind of independence criterion between the output label of classifier and the true label. The PA measure is indeed a dependence criterion due to minus RA from A, i.e., PA satisfies the property that its value is zero if and only if the output label of classifier and the true label are statistically independent.

For simplicity, we omit the functional dependence on X and Y in the following notations. Here, we give some representations of PA for further analysis.

1. For a learning task, the true label of instance is fixed. The partitions in $\mathcal{H}_{q(h)}$ are the possible predict vectors with the same class distribution. They have different accuracy values. The expectation is calculated over these values.

Definition 6. *The linear-fractional measure [28] is defined as*

$$\Psi(h(X), Y) = \frac{a_0 + a_1TP + a_2FP + a_3FN + a_4TN}{b_0 + b_1TP + b_2FP + b_3FN + b_4TN}, \tag{17}$$

where $a_i, b_i, i = 0, \ldots, 4$ are constants.

Due to $L(h) = 1 - A = FP + FN$ and $q(h) = p + FP - FN$, PA can be represented as

$$PA = 1 - \frac{L(h)}{p + (1-2p)q(h)}, \tag{18}$$

$$= 1 - \frac{FP + FN}{2p(1-p) + (1-2p)(FP - FN)}. \tag{19}$$

Obviously, PA belongs to the linear-fractional measures.

To give more intuition of PA, we show some learning settings that PA and A are linear.

Theorem 1. *PA and A are linear in the following cases: (1) when $p = \frac{1}{2}$, $PA = 2A - 1$; (2) when $q(h) = q_0$, q_0 is a constant taking value in [0,1], $PA = 1 - \frac{A}{p+(1-2p)q_0}$.*

We omit the proof of Theorem 1 because it can be easily obtained by definition.

The first case of Theorem 1 corresponds to a learning problem where the class distribution is totally balanced. The second case means the positive class probability of the classifier is fixed, which may occurs in the proportion learning problem [43]. In this problem, the training data is provided in groups and only the proportion of each class in each group is known. In such two cases, the evaluation results of PA and A are consistent, and PA can be maximized through maximizing A.

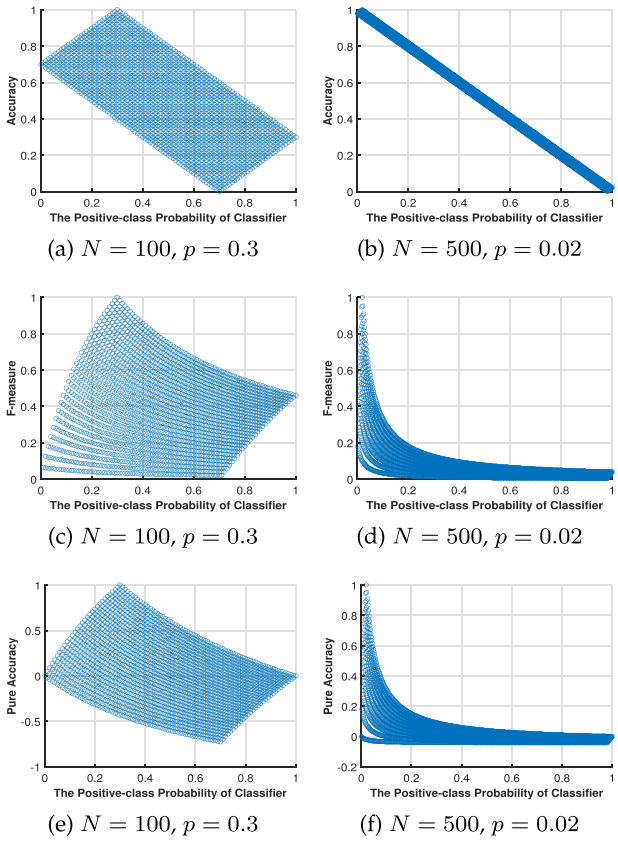
2.3 Advantages of Pure Accuracy

In [2], we have shown that compared with A, PA is class distribution insensitive and lower biased. As is well-known, F-measure (FM) is a kind of measure suited to learning [44]. Here, we aims to compare PA with A and FM in the view of class distribution insensitivity and discrimination.

Far more than the class imbalance of data, many factors can influence the class distribution of classifier, such as noise, overlap and even human-bias [45]. These factors often exists together in a learning task. For a learning task, the property of class distribution insensitivity helps the learning algorithm get closer to the true class distribution.

Advantage 1 (Class Distribution Insensitivity).

As the definition of RA, a binary partition can be deemed as an output of a classifier. By enumerating all possible binary partitions, we study how the value of A, FM and PA change with the class distribution of classifier. When $N = 100$, $p = 0.3$ and $N = 500$, $p = 0.02$, we calculate the values of the measures and the positive-class probability of the binary partition. Fig. 1 visualizes the calculation results. From Figs. 1a and 1b, it can be observed that the leftmost point is far higher than the rightmost point, indicating that the accuracy value plane is inclined. From Figs. 1c and 1d, it can be observed that the rightmost point is not equal to the leftmost point, indicating that the F-measure value plane is inclined. From Figs. 1e and 1f, it can be observed that both the leftmost point and the rightmost point are equal to zero, indicating that the pure accuracy plane is not inclined.



(a) $N = 100, p = 0.3$ (b) $N = 500, p = 0.02$

(c) $N = 100, p = 0.3$ (d) $N = 500, p = 0.02$

(e) $N = 100, p = 0.3$ (f) $N = 500, p = 0.02$

Fig. 1. Comparison on class distribution insensitivity.

These signifies that PA is a performance evaluation measure that is more insensitive to the class distribution of classifier than A and FM.

Next, we give some brief theoretical analysis on the class distribution insensitivity of PA and FM. It has been shown that the optimal rule of a a linear fractional measure has the formulation of $\eta(\boldsymbol{x}) > \rho$ [2], where $\eta(\boldsymbol{x}) = \mathbb{P}(y = 1|\boldsymbol{x})$ and $\rho \in [0, 1]$. In Theorem 1 of [2], it has been shown that for PA, the decision value $\rho_{PA} = \eta(\boldsymbol{x}) > (\frac{1}{2} - p)PA^* + p$, where $PA^* = PA(h^*_{PA})$, $h^*_{PA}(\boldsymbol{x}) = \arg\max_h PA(h)$ and $p = \mathbb{P}(Y = +1)$. By the same technique in proving Theorem 1 of [2], we obtain that for FM, the decision value is $\rho_{FM} = \frac{1}{2}FM^*$, where $FM^* = FM(h^*_{FM})$ and $h^*_{FM}(\boldsymbol{x}) = \arg\max_h FM(h)$.

Obviously, ρ_{PA} is a function related to p, while ρ_{FM} is not related to p. This signifies that the optimal rule learned by PA is able to tune with the class distribution of the true label. This makes PA not be biased to different classes. Thus PA is class distribution insensitive. However, from ρ_{FM}, we cannot obtain such observation.

Next, we will show that compared with FM, learning by PA is better.

Theorem 2. *For two classifiers $h_i, i = 1, 2$, satisfying*

$$\begin{cases} PA(h_1) \geq PA(h_2) \\ FM(h_1) \leq FM(h_2), \end{cases} \tag{20}$$

we have

$$\begin{cases} L(h_1) \leq L(h_2) \\ q(h_1) \leq q(h_2) \end{cases} \tag{21}$$

TABLE 2
An Example on Distinction Ability of a and PA

X	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	A	PA
Y	-1	-1	-1	-1	-1	-1	-1	-1	$+1$	$+1$		
h_1	-1	-1	-1	-1	-1	-1	$+1$	$+1$	$+1$	$+1$	0.8	0.5455
h_2	-1	-1	-1	-1	-1	-1	-1	$+1$	$+1$	-1	0.8	0.3750
h_3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.8	0

From Theorem 2, we can conclude that for a classifier with a bigger PA but a smaller FM, both the error probability and positive class probability are smaller. This signifies that when the evaluation results of PA and FM are inconsistent, the evaluation of PA is more reasonable because it rewards the classifier with a higher accuracy value and a lower probability to output the minority class.

Advantage 2 (More Discriminative). Accuracy is the expectation of $0-1$ step function. The value range of it is limited. As shown in Table 2, although h_1, h_2 and h_3 make different predictions on ten instances, they are assigned with the same A value. However, they have different PA values. Furthermore, we have the following result on the discrimination ability of PA and A.

Theorem 3. *Let $Table = (TP, FP, FN, TN)$ denote the confusion matrix. Suppose the number of instances is N, for two confusion matrices $Table_i, i = 1, 2$, let*

$$\begin{aligned} \mathcal{P} &= \{(Table_1, Table_2)|PA_1 \neq PA_2, A_1 = A_2\}, \\ \mathcal{Q} &= \{(Table_1, Table_2)|A_1 = A_2\}, \end{aligned} \tag{22}$$

we have $|\mathcal{P}| \to |\mathcal{Q}|$ as $N \to \infty$.

The set $|\mathcal{P}|$ contains the pair of classifiers that are assigned the same A value but different PA value. The set $|\mathcal{Q}|$ contains the pair of classifiers that are assigned the same A value. Theorem 3 tells us that with the number of instances tends to infinity, the size of $|\mathcal{P}|$ tends to the size of $|\mathcal{Q}|$, which signifies that PA can distinguish almost all classifier pairs that cannot be distinguished by A.

Next, we compare the distinction ability of PA and FM for good classifiers. A classifier is good if its bias is low and accuracy is high. The bias is defined as

$$\begin{aligned} Bias &= \left|\mathbb{P}(h(X) = +1|Y = -1) - \mathbb{P}(h(X) = -1|Y = +1)\right| \\ &= \left|\frac{FP}{1-p} - \frac{FN}{p}\right|, \end{aligned} \tag{23}$$

which measures the difference between the error probability of the two classes. Fig. 2 depicts three groups of bar when $N = 100$. From left to right, the bars are the number of pairs that h_1 is better than h_2, the number of better classifiers that PA can distinguish and the number of better classifiers that FM can distinguish, respectively. That is, the bars are the size of the set $\mathcal{B} = \{(Table_1, Table_2)|A_1 \geq A_2, Bias_1 \leq Bias_2\}$, the number of table pairs that satisfies $PA_1 \geq PA_2$ in $\mathcal{B}$ and the number of table pairs that satisfies $FM_1 \geq FM_2$ in $\mathcal{B}$, respectively. From Fig. 2, we observe that the bars of PA are higher than the ones of FM under different numbers of

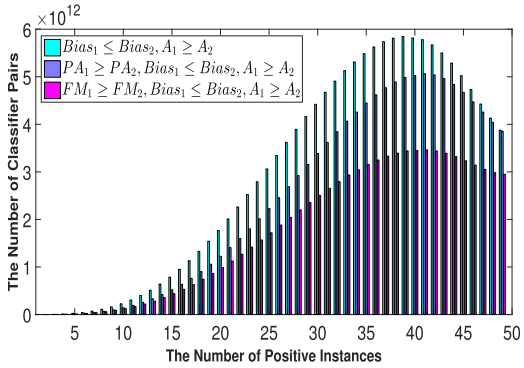


Fig. 2. Comparison on distinction ability.

positive instances. This means that PA has a better distinction ability for good classifiers than FM.

3 The Generalization Ability of Learning by PA

Through the training data set, classifiers can be obtained by optimizing some performance measures. How the classifiers perform on the new data is known as the generalization ability or the learning ability. The generalization ability investigation about PA can be summarized as bounding the deviation between the true PA and the empirical PA, which is

$$|PA(L,p,q) - \widehat{PA}_N(\widehat{L}_N, \widehat{p}_N, \widehat{q}_N)|. \tag{24}$$

The generalization bound is of great importance for us to understand what factors influence the performance. A tighter bound maybe helpful to design an algorithm with a fast speed of convergence and high generalization ability. Many effort have been put on developing tighter generalization bound [24], [46], [47], [48], [49], [50]. In this section, we investigate the learning ability of PA by giving a tighter generalization bound.

3.1 An Existing $O(1/\sqrt{N})$ Bound of PA

Concentration inequalities provide bounds on how a random variable deviates from its expected value. For simple random variables and their summations, these kind of inequalities have been thoroughly studied and well developed [51].

There are two obstacles of using concentration inequalities on the linear-fractional measures. One is that the empirical measure is not an unbiased estimation of the true measure. The other is that the linear-fractional measures cannot be obtained by summations on individual instances. To solve these obstacles, K. Dembczyński et.al. [32] employed the p-Lipschitz continuity, which is a generalized property of the Lipschitz continuity with replacing the Lipschitz constant by a parameter in terms of p.

Definition 7 (p-Lipschitz [32]). *For performance measure $\Psi(Z,p,q)$, it is p-Lipschitz with respect to Z,p,q. If for any feasible Z_i,p_i,q_i, $i=1,2$, it satisfies*

$$\begin{aligned}&\big|\Psi(Z_1,p_1,q_1) - \Psi(Z_2,p_2,q_2)\big| \\ &\quad \le Z_p\big|Z_1 - Z_2\big| + P_p|p_1-p_2| + Q_p|q_1-q_2|,\end{aligned} \tag{25}$$

where Z_p, P_p, Q_p are constants depended on p.

Thus, the upper bound of the deviation on Ψ can be obtained by combining the bounds of deviations on Z,p,q.

The complexity parameter of the hypothesis space is a key factor in developing a generalization bound. In [52], a more practical complexity parameter and a theoretical framework for deriving bounds based on it for preference-based learning are proposed. In [53], non-trivial bounds of some complexity parameters are determined, based on which the proposed algorithm was proved to be near-optimal. In binary classification, the Rademacher complexity is a good choice for infinite hypotheses space. Based on Rademacher complexity, one can obtain tighter bounds than based on the growth function and the VC-dimension [54]. Rademacher complexity measures the data description capability of the hypothesis space by measuring the fitting ability to the random uniform noise.

Definition 8 ([54]). *The Rademacher complexity of a hypothesis class $\mathcal{F}$ is defined as*

$$R(\mathcal{F}) = \mathbb{E}_{\sigma,X}\sup_{f\in\mathcal{F}}\left|\frac{2}{N}\sum_{i=1}^{N}\sigma_i f(x_i)\right|, \tag{26}$$

where σ_i are r.v. with $\mathbb{P}(\sigma_i=+1) = \mathbb{P}(\sigma_i=-1) = \frac{1}{2}$.

Based on the property of p-Lipschitz continuity and the Rademacher complexity, a generalization bound for performance measures is given [32].

Theorem 4 ([32]). *Let $\Psi(TP,p,q)$ be a p-Lipschiz continuous performance measure w.r.t TP,p,q. Then, for all $h\in\mathcal{H}$, $t>0$, with a probability at least $1-\exp\{-t\}$ over the random choice of sample $\mathcal{S}_N$, we have*

$$\begin{aligned}&\big|\Psi(TP,p,q) - \widehat{\Psi}_N(TP,p,q)\big| \\ &\le \max\{TP_p,P_p,Q_p\}\left(2R(\mathcal{H}) + 3\sqrt{\frac{t+\ln 4}{2\ N}} + \frac{1}{\sqrt{N}}\right).\end{aligned} \tag{27}$$

By simple calculation, we know that PA is p-Lipschitz w.r.t TP,p,q with the constant $\max\{TP_p,P_p,Q_p\} = 3/p$.

Corollary 1 (Corollary of Theorem 4). *For all $h\in\mathcal{H}$, $t>0$, with a probability at least $1-\exp\{-t\}$ over the random choice of sample $\mathcal{S}_N$, we have*

$$\begin{aligned}&\left|PA(y,h(\boldsymbol{x})) - \widehat{PA}_N(y,h(\boldsymbol{x}))\right| \\ &\qquad \le \frac{3}{p}\left(2R(\mathcal{H}) + 3\sqrt{\frac{t+\ln 4}{2\ N}} + \frac{1}{\sqrt{N}}\right).\end{aligned} \tag{28}$$

Generally, the order of the Rademacher complexity is $O(1/\sqrt{N})$ [50]. Corollary 1 gives a generalization bound of PA in an $O(1/\sqrt{N})$ order. From Corollary 1, we can draw a conclusion that the gap between the true PA value and the empirical one tends to zero with the instances tends to infinity and the rate of decline is $O(1/\sqrt{N})$.

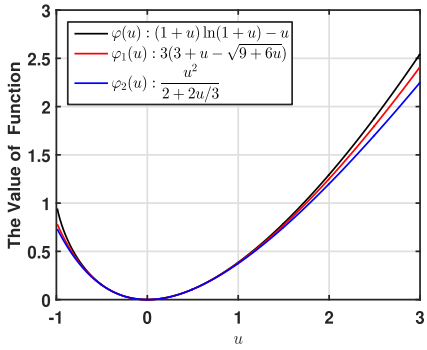


Fig. 3. Comparison of the Elementary Functions.

3.2 A Tighter Concentration Inequality for Bounding PA

The self-bounding property and a sub-poissonian concentration inequality are presented in this subsection. The property of self-bounding was first proposed by S. Boucheron [55]. With the self-bounding property, the variance of variables can be dominated by their expectations, which signifies that most of the distribution information is concentrated around the expectation. Popular complexity measures are self-bounding, such as the VC-dimension and the Rademacher complexity.

Definition 9 (Self-Bounding [55]). *For independent random variables $X_1,\ldots,X_N \in \mathcal{X}$, let $\overline{X} = (X_1,\ldots,X_N) \in \mathcal{X}^N$, and $\overline{X}^{(i)} = (X_1,\ldots X_{i-1},X_{i+1},\ldots,X_N) \in \mathcal{X}^{N-1}$ for each i. A measurable function $Z:\overline{X}\to\mathcal{R}$ satisfies the self-bounding property, if there exists a measurable function $Z_i:\overline{X}^{(i)}\to\mathcal{R}$, such that, for every $\overline{X}\in\mathcal{X}^N$, we have*

$$0 \le Z(\overline{X}) - Z_i(\overline{X}^{(i)}) \le 1, \tag{29}$$

$$\sum_{i=1}^{N}\Big(Z(\overline{X}) - Z_i(\overline{X}^{(i)})\Big) \le Z(\overline{X}). \tag{30}$$

The self-bounding property is more advanced than the commonly-used bounded difference. The bounded difference property is the foundation of the widely-used McDiarmid's inequality, which is the foundation of Theorem 4 and Corollary 1. The self-bounding facilitates the following tighter Sub-Poissonian concentration inequality.

Lemma 1 (A Sub-Poissonian Inequality [55]). *If Z is a self-bounding function, then for all $\varepsilon > 0$, we have*

$$\mathbb{P}(Z \ge \mathbb{E}Z + \varepsilon) \le \exp\left\{-\mathbb{E}Z\varphi\left(\frac{\varepsilon}{\mathbb{E}Z}\right)\right\}. \tag{31}$$

Moreover for $0 < \varepsilon < \mathbb{E}Z$ holds

$$\mathbb{P}(Z \le \mathbb{E}Z - \varepsilon) \le \exp\left\{-\mathbb{E}Z\varphi\left(\frac{\varepsilon}{\mathbb{E}Z}\right)\right\}, \tag{32}$$

where $\varphi(u) = (1+u)\ln(1+u) - u$, for $u \ge -1$.

Sub-Poissonian inequality gives a bound in terms of the expectation. The bound has no connection with the variance or some variance-like terms, which will be helpful to develop a tighter bound.

From the concentration inequality, we are more interested in estimating the deviation upper bound ε with a specified confidence level. But the inverse function of $\varphi(u)$ in Lemma 1 is not closed. This hinders the functional expression of the deviation bound with respect to the confidence level. To handle this, one often further amplifies the inequality by an elementary inequality [51]

$$\varphi(u) \ge \varphi_2(u) = \frac{u^2}{2 + 2\ u/3}.$$

Here, we use $\varphi_1(u)$ to amplify $\varphi(u)$

$$\varphi_1(u) = 3(3 + u - \sqrt{9 + 6\ u}).$$

The inverse function of $\varphi_1(u)$ is closed, and the proposed lower bound $\varphi_1(u)$ is larger than the usually used $\varphi_2(u)$. As shown in Fig. 3, $\varphi_1(u)$ locates between $\varphi_(u)$ and $\varphi_2(u)$. Based on $\varphi_1(u)$, we have

Corollary 2. *If Z is a self-bounding function and satisfies $\mathbb{E}Z > 0$, then for all $t > 0$, we have*

$$\mathbb{P}\left(Z \ge \mathbb{E}Z + \frac{t}{3} + \sqrt{2t\mathbb{E}Z}\right) \le \exp\{-t\}, \tag{33}$$

moreover for $t > 4\mathbb{E}Z$, we have

$$\mathbb{P}\left(Z \le \mathbb{E}Z - \frac{t}{3} - \sqrt{2t\mathbb{E}Z}\right) \le \exp\{-t\}. \tag{34}$$

Next, we extend Corollary 2 to general functions.

Lemma 2. *Let $\mathcal{F}:\mathcal{R}\to[0,1]$ be a non-zero measurable and additive [2] function class defined on random variables Z. Then, for all $f\in\mathcal{F}$, $t>0$, with a probability at least $1-\exp\{-t\}$ over the i.i.d. sample set $\mathcal{D}_N = \{z_1,\ldots,z_N\}$, we have*

$$\mathbb{E}f(z) \le \widehat{\mathbb{E}}_N f(z) + R(\mathcal{F}) + \frac{t}{3\ N} + \sqrt{\frac{2tR(\mathcal{F})}{N}}, \tag{35}$$

and

$$\widehat{\mathbb{E}}_N f(z) \le \mathbb{E}f(z) + R(\mathcal{F}) + \frac{t}{3\ N} + \sqrt{\frac{2tR(\mathcal{F})}{N}}. \tag{36}$$

Lemma 2 is a major result. It provides an upper and lower bound for the expectation of general additive functions. The bound is based on the Radermacher complexity $R(\mathcal{F})$, the number of instances N and the confidence level term t. Its order is $O(1/N + 1/\sqrt{N})$. Lemma 2 improves the bounds of Theorem 3.1 in [54].

Lemma 2 and its proof skills can be extended to reinforcement learning and stochastic convex optimization settings to improve the existing generalization bound, which maybe helpful to design algorithms with a fast convergence rates.

Based on Lemma 2, a generalization bound on the probability of error can be developed.

2. Here, the additive property means that $\widehat{\mathbb{E}}_N f(z) = \frac{1}{N}\sum_{i=1}^{N} f(z_i)$.

Theorem 5. *Suppose that ϕ_1, ϕ_2: $\mathcal{R} \to [0,1]$ satisfies: $\phi_2(u) \leq \mathbb{I}[u \leq 0] \leq \phi_1(u)$. For every $h \in \mathcal{H}$, $t > 0$, with a probability at least $1 - \exp\{-t\}$ over the random choice of sample $\mathcal{S}_N$, we have*

$$\mathbb{P}(yh(\boldsymbol{x}) \leq 0) \leq \widehat{\mathbb{E}}_N(\phi_1(yh(\boldsymbol{x})) + R(\tilde{\phi}_1 \circ \mathcal{H}) + \frac{t}{3\ N} + \sqrt{\frac{2tR(\tilde{\phi}_1 \circ \mathcal{H})}{N}}, \tag{37}$$

and

$$\mathbb{P}(yh(\boldsymbol{x}) \leq 0) \geq \widehat{\mathbb{E}}_N(\phi_2(yh(\boldsymbol{x})) - R(\tilde{\phi}_2 \circ \mathcal{H}) - \frac{t}{3\ N} - \sqrt{\frac{2tR(\tilde{\phi}_2 \circ \mathcal{H})}{N}}, \tag{38}$$

where
$\tilde{\phi}_i \circ \mathcal{H} = \{(\boldsymbol{x}, y) \to \phi_i(yh(\boldsymbol{x})) - \phi_i(0), h \in \mathcal{H}, i = 1, 2\}$.

Theorem 5 provides an upper and lower bound for the expectation of the error probability. The bound is based on some empirical loss ϕ, the Rademacher complexity $R(\mathcal{F})$, the number of instances N and the confidence level term t. Its order is $O(1/N + 1/\sqrt{N})$. Theorem 5 improves the bounds of Theorem 7 in [22].

3.3 A Tighter Generalization Bound of PA

Generalization bounds for PA is based on the above preliminary results, including Corollary 2, Lemma 2 and Theorem 5. First, we use the p-Lipschitz continuous property to handle PA.

Lemma 3. *PA is p-Lipschitz continuous with respect to L, p, q. That is, for $\forall L_i, p_i, q_i, i = 1, 2$, we have*

$$\left|PA(L_1, p_1, q_1) - PA(L_2, p_2, q_2)\right| \tag{39}$$

$$\leq \frac{2}{p}\left(\left|L_1 - L_2\right| + \left|p_1 - p_2\right| + \left|q_1 - q_2\right|\right), \tag{40}$$

where L is the error probability and p, q is the positive-class probability of the true label and the classifier, respectively.

Theorem 6. *For every $h \in \mathcal{H}$ and a large enough $t > 0$, with a probability at least $1 - \exp\{-t\}$, we have*

$$\left|PA(y, h(\boldsymbol{x})) - \widehat{PA}_N(y, h(\boldsymbol{x}))\right| \leq \frac{4}{p}\left(R(\mathcal{H}) + \frac{t + \ln 6}{2\ N} + \sqrt{\frac{2(t + \ln 6)}{N}}\left(\sqrt{R(\mathcal{H})} + \frac{\sqrt{p}}{2}\right)\right). \tag{41}$$

Theorem 6 gives a generalization bound of PA in an $O(1/N + 1/\sqrt{N})$ order. The bound is based on the Rademacher complexity, which is suited to both finite and infinite hypothesis space. From Theorem 6, we can draw a conclusion that the gap between the true PA value and the empirical ones tends to zero with the instances tends to infinity and the rate of decline is $O(1/N + 1/\sqrt{N})$. Theorem 6 considers the same factors as Corollary 1. Next, we show that with a large number of instances, the bound in Theorem 6 is tighter than the existing bound in Corollary 1.

Corollary 3. *Using $B_1(N)$ to denote the bound in Corollary 1 and $B_2(N)$ denote the bound in Theorem 6, i.e.,*

$$B_1(N) = \frac{3}{p}\left(2R(\mathcal{H}) + 3\sqrt{\frac{t_1}{N}} + \frac{1}{\sqrt{N}}\right), \tag{42}$$

and

$$B_2(N) = \frac{2}{p}\left(2R(\mathcal{H}) + \frac{t_2}{N} + \sqrt{\frac{2t_2}{N}}\left(2\sqrt{R(\mathcal{H})} + \sqrt{p}\right)\right), \tag{43}$$

where $t_1 = \frac{t + \ln 4}{2}$, $t_2 = t + \ln 6$, $R(\mathcal{H})$ is the Rademacher complexity of the hypothesis space $\mathcal{H}$, N is the number of instances, p is the probability of positive class and t is a term in confidence level. If

$$\sqrt{N} > \frac{t_2}{3\sqrt{t_1} + 1 - \sqrt{t_2}(2\sqrt{R(\mathcal{H})} + \sqrt{p})}, \tag{44}$$

we have that $B_1(N) > \frac{3}{2}B_2(N)$.

The bounds in Corollary 3 are based on the Rademacher complexity, which are suited to both finite and infinite hypothesis space. However, for some infinite hypotheses, the computation of Rademacher complexity is hard. To visually compare the bounds, we further amplify the Rademacher complexity by Massart's Lemma [54]. Massart's Lemma bounds the Rademacher complexity by the size of the hypothesis. Thus, the bounds can be calculated and depicted for some finite hypothesis space. Let $\gamma_{max} = \max_{f(\boldsymbol{x}_i)}(\sum_{i=1}^{N} f(\boldsymbol{x}_i)^2)^{1/2}$, Massart's Lemma tells us that the Rademacher complexity can be bounded by

$$R(\mathcal{F}) \leq \frac{\gamma_{max}\sqrt{2\log|\mathcal{F}|}}{N}, \tag{45}$$

where $|\mathcal{F}|$ is the size of $\mathcal{F}$.

Now, we compare the tightness of the bounds in a simple weighted ensemble learning scene.

Given T base classifiers $h_1, h_2, \ldots, h_T$, for the weighted ensemble learning, the prediction of an instance can be formulated as $h_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{j=1}^{T} w_j h_j(\boldsymbol{x})$. Here, we consider a simple weighted ensemble scene, where the weight of each base classifier takes value from $w \in \{-1, 0, 1\}$. The hypothesis space of this weight ensemble learning is $H_{\boldsymbol{w}} = \{\boldsymbol{w} | \boldsymbol{w} = (w_1, w_2, \ldots, w_T), w_i \in \{-1, 0, 1\}, 0 \leq i \leq T\}$. For T classifiers, we have $|H_{\boldsymbol{w}}| = 3^T$ because the weight of each classifier has three values.

To depict the bounds, we set $p = 0.3$, $t = \ln(1/0.95)$, $\gamma_{max} = \sqrt{N}$. Thus

$$R(\mathcal{H}_{\boldsymbol{w}}) \leq \frac{\gamma_{max}\sqrt{2T\log 3}}{N} = \frac{\sqrt{2T\log 3}}{\sqrt{N}}. \tag{46}$$

Putting the above size-based upper bound of Rademacher complexity in $B_i(N)$, we can obtain the upper bound of $B_i(N)$ and denote the upper bound as $\bar{B}_i(N)$, $i = 1, 2$. As shown in Fig. 4, We can see that in this simple weighted ensemble learning scene, $\bar{B}_2(N)$ (the red curve) falls faster than $\bar{B}_1(N)$(the blue curve).

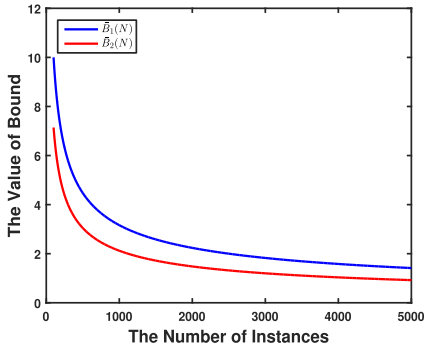


Fig. 4. Comparison of $\bar{B}_1(N)$ and $\bar{B}_2(N)$in the simple weighted ensemble learning scene.

4 Selective Ensemble Learning Based on PA

In this section, we design a learning algorithm that applies to any learning setting using a linear combination model of the input as the decision function. Such settings include the traditional classification which uses a linear combination of the features as the decision function, the kernel method which uses a linear combination of the features as the decision function, and the selective ensemble learning which uses a combination function of the base classifiers as the decision function.

In this paper, we use the algorithm in the setting of selective ensemble learning. The reason is that the selective ensemble learning does not contain too many learning tricks, such as kernel functions or data normalization. Thus, compared with other settings, it can highlight the role of performance measures.

In the scene of optimization based selective ensemble learning, the combination terms are the base classifiers. Let the base classifier set be $H = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_T\}$, where $\boldsymbol{h}_j = (h_j(\boldsymbol{x}_1), \ldots, h_j(\boldsymbol{x}_N))$. Let the weight vector be $\boldsymbol{w} = (w_1, w_2, \ldots, w_T) \in \mathcal{R}^+$. Following the traditional setting, the classifiers are combined by the weighted vote $h_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{j=1}^{T} w_j h_j(\boldsymbol{x})$ during training, and the prediction label for $\boldsymbol{x}$ is $+1$ if $h_{\boldsymbol{w}}(\boldsymbol{x}) > 0$; otherwise, the output is -1. With optimizing some specified performance measure, the optimal weight vector $\boldsymbol{w}^*$ is obtained, and the classifiers with a weight above a threshold will be selected.

In past decades, there are several optimization methods to select classifiers. In the following, we list two representative methods which optimized accuracy related measures:

- GASEN [18] significantly improved the performance of neural network ensemble through selective learning, which employed the empirical error probability as the performance measure

$$\frac{1}{N}\sum_{i=1}^{N}\left(\mathbb{I}[y_i h_{\boldsymbol{w}}(\boldsymbol{x}_i) < 0] + \frac{1}{2}\mathbb{I}[y_i h_{\boldsymbol{w}}(\boldsymbol{x}_i) = 0]\right). \tag{47}$$

  The optimal weights is found by the genetic algorithm.
- RSE [56] solved the problem of selective ensemble under the regularization framework. The objective function is the hinge loss function with a graph Laplacian regularizer. The hinge loss function is

$$\frac{1}{N}\sum_{i=1}^{N}\max\{0, 1 - y_i h_{\boldsymbol{w}}(\boldsymbol{x}_i)\}. \tag{48}$$

  The optimal weights is found by the quadratic program.

In this paper, we aim at developing a selective method based on PA. The linear learning algorithm optimizing PA can be formalized as

$$\min_{\boldsymbol{w}} \frac{\hat{L}_N(\boldsymbol{w})}{\hat{p}_N + (1 - 2\hat{p}_N)\hat{q}_N(\boldsymbol{w})}, \quad s.t. \quad \boldsymbol{w} > 0, \tag{49}$$

where $\hat{L}_N(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[y_i h_{\boldsymbol{w}}(\boldsymbol{x}_i) < 0]$ and $\hat{q}_N(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[h_{\boldsymbol{w}}(\boldsymbol{x}_i) > 0]$. This is a model of optimizing the non-convex linear-fractional measures Ψ (defined in Definition 6). Traditional gradient methods can not obtain a satisfactory solution for these measures. Besides, both the numerator and denominator of the objective function contain the non-smooth indicator function. Directly optimizing it leads to an NP-hard combinatorial problem. Thus, first, we need to approach the indicator function by some smooth functions. Here, we use the sigmoid function. Particularly, $\hat{L}_N(\boldsymbol{w})$ and $\hat{q}_N(\boldsymbol{w})$ are substituted by

$$\hat{L}_{N,s}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{1 + e^{y_i h_{\boldsymbol{w}}(\boldsymbol{x}_i)}}, \tag{50}$$

$$\hat{q}_{N,s}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{1 + e^{-h_{\boldsymbol{w}}(\boldsymbol{x}_i)}}. \tag{51}$$

As is well known, there are many convex functions that can be used to surrogate the indicator function, such as the hinge function, the exponential function, the logarithmic function and so on [16]. We choice the sigmoid function due to its boundness property. This property will be helpful for optimization.

4.1 Reviews on the Models of Optimizing Linear-Fractional Measures

Now we make a review on the existing models of optimizing the linear-fractional measures, including the plug-in method, SVMperf [37], the bisection method [31] and the gradient method [38].

4.1.1 Plug-In

The plug-in rule [28] refers to the rule with a formulation of $h(\boldsymbol{x}) = sign(\hat{\eta}(\boldsymbol{x}) - \delta^*)$, where $\hat{\eta}(\boldsymbol{x})$ is an estimator of the posterior probability $\eta(\boldsymbol{x}) = \mathbb{P}(Y = +1|X = \boldsymbol{x})$ and δ^* is a threshold. The plug-in method is a two-step method, which includes learning $\hat{\eta}(\boldsymbol{x})$ through minimizing a proper loss function and searching δ^* through maximizing the empirical fractional-measure. This method requires to estimate the posterior probability and to learn a proper threshold. Thus, it needs more data to avoid over-fitting.

4.1.2 Svmperf

The basis idea of SVMperf [37] is that the margin deviation between the true label vector and the others should be lager than their Ψ value

$$\min_{\boldsymbol{w},\xi\geq 0} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\xi \tag{52}$$

$$s.t. \quad \forall \bar{y}' \in \overline{\mathcal{Y}}\backslash\bar{y}: \quad \bar{y}h_{\boldsymbol{w}}^T(\boldsymbol{x}) - \bar{y}'h_{\boldsymbol{w}}^T(\boldsymbol{x}) \geq \Psi(\bar{y},\bar{y}') - \xi, \tag{53}$$

where $h_{\boldsymbol{w}}(\boldsymbol{x})$ is the prediction result, $\overline{\mathcal{Y}} = \{\pm 1\}^N$ is the space of label vector and $\bar{y} = (y_1, y_2, \ldots, y_N)$ is the true label vector. The time consuming of SVMperf is intolerable, because the number of constraints is $2^N - 1$. T. Joachims [37] iteratively solved the program with a sparse subset of the constraints set and the most violative constraint is added in each iteration.

4.1.3 Bisection

In the area of fractional program, there exists a method to solve the linear-fractional optimization problem by introducing a parameter.

Theorem 7 (Parameter-Based Method [57], [58]). *For program with a ratio of two functions as the objective function*

$$\min_{\boldsymbol{w}} \frac{F_1(\boldsymbol{w})}{F_2(\boldsymbol{w})}, \quad s.t. \quad \boldsymbol{w} \in \mathcal{W}, \tag{54}$$

where $F_2(\boldsymbol{w}) > 0$, the optimal solution can be obtained by the following program

$$\min_{\boldsymbol{w}} F_1(\boldsymbol{w}) - \lambda^* F_2(\boldsymbol{w}), \tag{55}$$

where λ^* is the zero root of

$$f(\lambda) = \min_{\boldsymbol{w}} F_1(\boldsymbol{w}) - \lambda F_2(\boldsymbol{w}). \tag{56}$$

Theorem 7 implies that the fractional measure can be equivalently transformed to a linear one $f(\lambda)$. The bisection method [31] is based on this idea. First, it learns a posteriori probability $\hat{\eta}(\boldsymbol{x})$. In each iteration, with the current λ, the cost matrix is updated according to the function $f(\lambda)$. Then, based on $\hat{\eta}(\boldsymbol{x})$ and $f(\lambda)$, the classifier point-wisely outputs the label that minimizes the posteriori cost-sensitive loss. The optimal λ is searched in binary.

4.1.4 Gradient Method

The gradient method [38], [39] is an extension of the SVMperf method. Hazan et.al. [38] proved that in the binary classification, the gradient of the linear-fractional measure Ψ is

$$\nabla_{\boldsymbol{w}}\Psi(\bar{y},\bar{y}_{\boldsymbol{w}}) = \frac{1}{\epsilon}\left(\sum_{i=1}^{N}\boldsymbol{x}_i\bar{y}_{loss,i} - \sum_{i=1}^{N}\boldsymbol{x}_i\bar{y}_{\boldsymbol{w},i}\right), \tag{57}$$

where the two label vectors $\bar{y}_{\boldsymbol{w}}$ and $\bar{y}_{loss}$ are

$$\bar{y}_{\boldsymbol{w}} = \arg\max_{\bar{y}'\in\mathcal{Y}} \bar{y}'h_{\boldsymbol{w}}^T(\boldsymbol{x}), \tag{58}$$

$$\bar{y}_{loss} = \arg\max_{\bar{y}'\in\mathcal{Y}} \{\Psi(\bar{y},\bar{y}') + \epsilon\bar{y}'h_{\boldsymbol{w}}^T(\boldsymbol{x})\}, \tag{59}$$

respectively. $\bar{y}_{loss}$ is exactly the label vector that most violates the constraint in SVMperf. With the proved gradient, the weight vector can be updated by

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \lambda\nabla_{\boldsymbol{w}}\Psi(\bar{y},\bar{y}_{\boldsymbol{w}}), \tag{60}$$

where λ is the update step.

4.2 PASE: Pure Accuracy Based Selective Ensemble Learning

It seems that the model (49) can be easily solved by a divide-and-conquer strategy. Namely, one can fix the class distribution $\hat{q}_{N,s}(\boldsymbol{w})$ at a constant q_0. Then, with the linear constraint, an optimal solution can be found by optimizing the $\hat{L}_{N,s}(\boldsymbol{w})$. Through varying $\hat{q}_{N,s}(\boldsymbol{w})$ at all feasible constants, one can obtain multiple solutions and fetch the optimal one among them. This strategy is natural and straightforward, while it is too time-consuming and the feasible range of q_0 is difficult to determine. Here, we introduce the following theorem to solve the model

Theorem 8 ([59]). *For program with a ratio of two positive functions as the objective function*

$$\min_{\boldsymbol{w}} \frac{F_1(\boldsymbol{w})}{F_2(\boldsymbol{w})}, \quad s.t. \quad \boldsymbol{w} \in \mathcal{W}. \tag{61}$$

Let

$$\boldsymbol{w}(r) = \arg\max_{\boldsymbol{w}\in\mathcal{W}} \frac{F_1(\boldsymbol{w})}{r}, \quad s.t. F_2(\boldsymbol{w}) \geq r. \tag{62}$$

The optimal solution of (61) can be obtained by $\boldsymbol{w}(r^*)$ if and only if r^* is the optimal solution of:

$$\min_r \frac{F_1(\boldsymbol{w}(r))}{r} \quad s.t. \min_{\boldsymbol{w}\in\mathcal{W}} F_2(\boldsymbol{w}) \leq r \leq \max_{\boldsymbol{w}\in\mathcal{W}} F_2(\boldsymbol{w}), \tag{63}$$

It is easy to solve program (62). The difficulty of program (63) is that $F_1(\boldsymbol{w}(r))$ is an implicit function the variable r. R. W. Freund [59] provided two sophisticated linear support functions on $F_1(\boldsymbol{w}(r))/r$ and adaptively solving the the optimal r in different divisions. Particularly, in each interval $[r^{(i)}, r^{(i+1)}]$, $0 \leq r^{(i)} \leq r^{(i+1)}$, let

$$F(r) = \min_{\boldsymbol{w}} \left\{ \frac{F_1(\boldsymbol{w})}{r} \middle| F_2(\boldsymbol{w}) \geq r, \boldsymbol{w} \in \mathcal{W} \right\}, \tag{64}$$

$$\widetilde{F}_i(r) = \min_{\boldsymbol{w}} \left\{ \frac{F_1(\boldsymbol{w})}{r^{(i+1)}} \middle| F_2(\boldsymbol{w}) \geq r, \boldsymbol{w} \in \mathcal{W} \right\}, \tag{65}$$

then R. W. Freund [59] proved that

$$\frac{F_1(\boldsymbol{w}(r))}{r} \geq F(r^{(i)}) + F_i'(r - r^{(i)}) = \underline{F(r)}, \tag{66}$$

$$\frac{F_1(\boldsymbol{w}(r))}{r} \geq F(r^{(i+1)}) + \lambda_F(r - r^{(i+1)}), \tag{67}$$

where

$$F_i' = \frac{r^{(i+1)}}{r^{(i)}} \frac{\widetilde{F}_i(r^{(i)}) - F(r^{(i)})}{r^{(i+1)} - r^{(i)}},$$

and λ_F is the Lagrange multiple of the program $\widetilde{F}_i(r^{(i+1)})$.

Here, we use the right-hand support function to approach $F_1(\boldsymbol{w}(r))/r$ and the optimal r^* will be searched by finding the minimizer of $F(r)$ from the intervals.

For searching r^*, we use the a general method in solving One-Dimensional Global Optimization (ODGO) [59], [60]. For completeness, we present ODGO in Algorithm 1.

Algorithm 1. ODGO

INPUT: The fractional measure F_1/F_2

PROCEDURE:

Determine $r^{(1)} = \min_{\boldsymbol{w}} F_2(\boldsymbol{w})$, $r^{(2)} = \max_{\boldsymbol{w}} F_2(\boldsymbol{w})$

Initialize the index $i = 1$, $r_{list} = \{r^{(1)}, r^{(2)}\}$, $\tilde{r}_{list} = \{1\}$, $\underline{\widetilde{F}}_{list} = \{1\}$

while $|r^{(i)} - r^{(i+1)}| > \epsilon$ **do**

Set $\hat{r} = (r^{(i)} + r^{(i+1)})/2$

Solve the programs

$$\underline{\widetilde{F}}_L = \min_{r\in[r^{(i)},\hat{r}]} \underline{F(r)}, \quad \underline{\widetilde{F}}_R = \min_{r\in[\hat{r},r^{(i+1)}]} \underline{F(r)},$$

and obtain the solution $\tilde{r}_L$ and $\tilde{r}_R$, respectively

Update $r_{list} = \{r^{(1)}, \ldots, r^{(i)}, \hat{r}, r^{(i+1)}, \ldots, r^{(t)}\}$

Update

$$\underline{\widetilde{F}}_{list} = \{\underline{\widetilde{F}}^{(1)}, \ldots, \underline{\widetilde{F}}^{(i-1)}, \underline{\widetilde{F}}_L, \underline{\widetilde{F}}_R, \underline{\widetilde{F}}^{(i+1)}, \ldots, \underline{\widetilde{F}}^{(t)}\}$$

Update

$$\tilde{r}_{list} = \{\tilde{r}^{(1)}, \ldots, \tilde{r}^{(i-1)}, \tilde{r}_L, \tilde{r}_R, \tilde{r}^{(i+1)}, \ldots, \tilde{r}^{(t)}\}$$

Find the index $i = \arg\min_j \{\underline{\widetilde{F}}^{(j)}, 1 \le j \le t+1\}$

end while

$r^* = \tilde{r}^{(i)}$

OUTPUT: r^*

Now, we present the selective algorithm based on optimizing PA in Algorithm 2 and name it as PASE. The Step 2-Step3 of PASE finds the optimal weight vector which maximizes the pure accuracy, then the classifiers with a weight above the average value will be preserved. In prediction, the selected classifiers make a majority vote for the test data. That is, the test data will be classified into the class that is assigned by most of the selected classifiers.

Algorithm 2. PASE

INPUT: The training data S_N, classifier set $H = \{h_1, h_2, \ldots, h_T\}$, where $h_j = (h_j(\boldsymbol{x}_1), \ldots, h_j(\boldsymbol{x}_N))^T$.

PROCEDURE:

Initialize $\boldsymbol{w}^0 = (0, 0, \ldots, 0)$.

Solve r^* by Algorithm 1.

Solve $\boldsymbol{w}^*$ by the interior-point method

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \frac{2}{r^*} \hat{L}_{N,s}(\boldsymbol{w}),$$

$$s.t. (1 - 2\hat{p}_N)\hat{q}_{N,s}(\boldsymbol{w}) \ge r^* - \hat{p}_N.$$

OUTPUT: The model parameter $\boldsymbol{w}^*$ and the set $H^* = \{h_j | w_j^* > \frac{1}{T}\sum_{k=1}^{T} w_k^*, j = 1, 2, \ldots T\}$

Absolutely, Algorithm 2 can also be used to build linear classifiers with replacing the classifiers set with the feature set and it can also be extended to kernel method with replacing the classifiers set with the kernel matrix.

Both the bisection method and PASE turn the fractional program to an easy-to-solve program with respect to the model parameters (termed as the master program) and a one-dimensional program with respect to an introduced parameter. Their differences are

- The bisection method is based on Theorem 7, and PASE is based on Theorem 8. The master program of Theorem 7 uses the introduced parameter to trade off the numerator and the denominator, while PASE uses the introduced parameter to substitute the denominator. Thus, the bisection is finished by minimizing a cost-sensitive loss, while Algorithm 2 is finished by directly optimizing the objective measure after adaptively finding the optimal denominator.
- In solving the one-dimensional program, the bisection method uses the binary search method, while PASE is based on Algorithm 1. The binary method searches in one direction and is easy to find a local optimal solution; however, Algorithm 1 stores the optimal solution from both left and right directions in each iteration, which will be helpful to find a global optimal solution.
- Due to Theorem 8, PASE can be easily extended to the models with some regularization terms, such as the SVM model with maximizing the linear-fractional measures, while the bisection method cannot.

TABLE 3
Data Sets Description

ID	Name	Objects	Attributes	IR
1	Crx	653	15	1:1.21
2	Australian	690	14	1:1.25
3	MicroMass	931	1300	1:1.59
4	Wdbc	569	30	1:1.68
5	Bands	365	19	1:1.70
6	Ionoshpere	351	33	1:1.79
7	Pima	768	8	1:1.87
8	Titanic	2201	3	1:2.10
9	German	1000	20	1:2.33
10	Parkinson's Disease	756	752	1:2.94
11	Segment	2310	19	1:6.02
12	Dermatology	366	34	1:16.9
13	DrivFace	606	6400	1:21.45
14	Winequality-red-4	1599	11	1:29.17
15	Wine-White-3-9-VS-5	1482	11	1:58.28
16	Abalone-20-vs-8-9-10	1916	8	1:72.69
17	Scene	738	1000	1:1.03
18	Butterflies	176	1000	1:3.19
19	Animals	234	1000	1:5.88
20	Vehicles	888	1000	1:9.10

4.3 Experimental Comparison With Selective Ensemble Learning Algorithms

In this subsection, the performance of PASE is verified on sixteen benchmark data sets and four image data sets. Their information is presented in Table 3, in which MicroMass, Parkinson's Disease and DrivFace are from UCI [61] and the remaining sets are from KEEL [62].

TABLE 4
PA Value of Selective Ensemble

	Pure Accuracy						
Data ID	DT	GASEN	RSE	SVMperf-Gmean	Bisection-FM	Gradient-BA	PASE
1	0.6685 ± 0.0227 •	0.7270 ± 0.0384	0.7244 ± 0.0379	0.7264 ± 0.0350	0.7251 ± 0.0384	0.7167 ± 0.0316 •	**0.7375** ± 0.0405
2	0.6460 ± 0.0311 •	0.7134 ± 0.0472	0.7185 ± 0.0535	0.7160 ± 0.0473	**0.7227** ± 0.0544	0.7130 ± 0.0475	0.7220 ± 0.0472
3	0.7145 ± 0.0207 •	0.8772 ± 0.0295 •	0.8823 ± 0.0299 •	0.8754 ± 0.0394 •	0.8924 ± 0.0281	0.8789 ± 0.0319 •	**0.8968** ± 0.0208
4	0.8374 ± 0.0217 •	0.9032 ± 0.0299	0.9035 ± 0.0332	0.9072 ± 0.0343	0.9039 ± 0.0413	0.9082 ± 0.0347	**0.9098** ± 0.0337
5	0.1522 ± 0.0264 •	0.2745 ± 0.0999 •	0.2246 ± 0.0604 •	0.2433 ± 0.0915 •	0.2891 ± 0.0934 •	0.2431 ± 0.0827 •	**0.3237** ± 0.0930
6	0.7360 ± 0.0338 •	0.8307 ± 0.0550	0.8239 ± 0.0519	**0.8388** ± 0.0539	0.8271 ± 0.0549	0.8380 ± 0.0538	0.8325 ± 0.0524
7	0.3596 ± 0.0309 •	0.4465 ± 0.0624 •	0.4309 ± 0.0532 •	0.4498 ± 0.0562 •	0.4553 ± 0.0576	0.4438 ± 0.0568 •	**0.4741** ± 0.0604
8	0.4092 ± 0.0298 •	0.4055 ± 0.0307 •	0.3997 ± 0.0351 •	0.4059 ± 0.0343 •	0.4121 ± 0.0420 •	0.3934 ± 0.0392 •	**0.4277** ± 0.0441
9	0.2444 ± 0.0138 •	0.3184 ± 0.0433 •	0.2847 ± 0.0389 •	0.2988 ± 0.0449 •	0.3388 ± 0.0592 •	0.2814 ± 0.0452 •	**0.3825** ± 0.0499
10	0.4044 ± 0.0280 •	0.5887 ± 0.0709	0.5643 ± 0.0703 •	0.6085 ± 0.0532	**0.6095** ± 0.0614	0.5972 ± 0.0486	0.6070 ± 0.0607
11	0.9561 ± 0.0098 •	0.9721 ± 0.0137	0.9683 ± 0.0178	0.9728 ± 0.0126	0.9748 ± 0.0118	0.9726 ± 0.0123	**0.9752** ± 0.0121
12	0.9286 ± 0.0590 •	0.9558 ± 0.0865 •	0.8768 ± 0.0921 •	0.9550 ± 0.0921	0.9585 ± 0.0713	0.9634 ± 0.0858	**0.9836** ± 0.0334
13	0.6021 ± 0.0513 •	0.7549 ± 0.1086	0.7020 ± 0.1432	0.7329 ± 0.1123	0.7393 ± 0.1071	0.7261 ± 0.1133	**0.7598** ± 0.0937
14	0.8749 ± 0.0191 •	0.9239 ± 0.0291	0.9234 ± 0.0315	0.9210 ± 0.0312	**0.9267** ± 0.0305	0.9210 ± 0.0311	0.9247 ± 0.0281
15	**0.0598** ± 0.0256 •	-0.0003 ± 0.0160	0.0073 ± 0.0323	0.0112 ± 0.0440	0.0205 ± 0.0518	0.0162 ± 0.0474	0.0354 ± 0.0646
16	0.1155 ± 0.0710	0.1110 ± 0.1188	0.0964 ± 0.1152	0.1242 ± 0.1214	0.1270 ± 0.1338	0.1290 ± 0.1256	**0.1435** ± 0.1248
17	0.1851 ± 0.0795	0.1488 ± 0.1317	0.1016 ± 0.1408 •	0.1260 ± 0.1326 •	0.1658 ± 0.1587	0.1226 ± 0.1256 •	**0.2080** ± 0.1771
18	0.8788 ± 0.0160 •	0.9160 ± 0.0235	**0.9200** ± 0.0226	0.9109 ± 0.0211	0.9193 ± 0.0228	0.9080 ± 0.0204 •	0.9182 ± 0.0230
19	0.8519 ± 0.0283 •	0.9686 ± 0.0427	0.9636 ± 0.0501	0.9477 ± 0.0611	0.9572 ± 0.0501	0.9454 ± 0.0621	**0.9688** ± 0.0367
20	0.7434 ± 0.0521 •	0.9019 ± 0.0838 •	0.8181 ± 0.1303 •	0.9097 ± 0.0768	0.9254 ± 0.0787	0.9123 ± 0.0721	**0.9282** ± 0.0738
Ave. Rank	6.1000	4.2000	5.2000	4.0000	2.5500	4.5000	1.4500

The image classification tasks are Scene, Butterflies, Animals and Vehicles data set. Please refer to the supplementary material for more detailed descriptions, available online. We extract vectorized features from each image in the data set with the pre-trained VGG-16 convolutional neural network [63]. Following the structure of VGG-16, the image is scaled to 224×224. The 1000-dimensional output of the final fully pooling layer is obtained as the pre-extracted feature vector.

Each data set is randomly divided into a training set and a test set at a ratio of 7:3. On each division, we run every method 30 times to evaluate the average performance. The performance is evaluated in terms of pure accuracy, TP and accuracy.

The base classifier is the binary decision tree (abbreviate as DT). The set of trees $H = \{h_1, h_2, \ldots, h_T\}$ is generated by bagging technical. In bagging, T samples are randomly sampled from the training data with replacement, then each tree is constructed on each sample. T is set up at 101.

The plug-in rule does not output a weight vector that optimizes the linear-fractional measure, and thus can not be used to select classifiers. We compare PASE with GASEN, RSE, SVMperf, bisection, gradient. The objective measure of SVMperf is Gmean, the objective measure of

TABLE 5
TP Value of Selective Ensemble

	TP						
Data ID	DT	GASEN	RSE	SVMperf-Gmean	Bisection-FM	Gradient-BA	PASE
1	0.3703 ± 0.0141 •	0.3969 ± 0.0187	0.3949 ± 0.0206	**0.3984** ± 0.0207	0.3949 ± 0.0231	0.3978 ± 0.0241	0.3984 ± 0.0217
2	0.3492 ± 0.0167 •	0.3729 ± 0.0235	0.3718 ± 0.0248 •	0.3726 ± 0.0220 •	0.3774 ± 0.0228	0.3704 ± 0.0249 •	**0.3814** ± 0.0238
3	0.3172 ± 0.0098 •	0.3452 ± 0.0118 •	0.3406 ± 0.0140 •	0.3457 ± 0.0153 •	0.3505 ± 0.0111	0.3471 ± 0.0125 •	**0.3524** ± 0.0108
4	0.3391 ± 0.0072 •	0.3516 ± 0.0100	0.3495 ± 0.0097 •	0.3520 ± 0.0086	0.3532 ± 0.0109	0.3525 ± 0.0096	**0.3535** ± 0.0099
5	0.1543 ± 0.0123 •	0.1284 ± 0.0313 •	0.0925 ± 0.0234 •	0.1171 ± 0.0337 •	0.1589 ± 0.0504 •	0.1101 ± 0.0321 •	**0.1706** ± 0.0330
6	0.2965 ± 0.0097 •	0.3177 ± 0.0165	0.3124 ± 0.0163 •	0.3154 ± 0.0166	0.3162 ± 0.0165	0.3154 ± 0.0166	**0.3185** ± 0.0148
7	0.1919 ± 0.0109 •	0.1981 ± 0.0217 •	0.1875 ± 0.0185 •	0.1986 ± 0.0191 •	0.2071 ± 0.0184 •	0.1960 ± 0.0212 •	**0.2266** ± 0.0179
8	0.1277 ± 0.0126 •	0.1185 ± 0.0171 •	0.1202 ± 0.0211 •	0.1208 ± 0.0214 •	0.1282 ± 0.0261 •	0.1161 ± 0.0219 •	**0.1404** ± 0.0271
9	0.1215 ± 0.0067 •	0.1068 ± 0.0163 •	0.0911 ± 0.0131 •	0.1043 ± 0.0174 •	0.1207 ± 0.0223 •	0.0961 ± 0.0205 •	**0.1556** ± 0.0280
10	0.1342 ± 0.0088 •	0.1420 ± 0.0225	0.1313 ± 0.0201 •	**0.1478** ± 0.0167	0.1452 ± 0.0195	0.1452 ± 0.0154	0.1459 ± 0.0203
11	0.1378 ± 0.0024 •	0.1388 ± 0.0025	0.1377 ± 0.0038 •	0.1390 ± 0.0023	**0.1396** ± 0.0023	0.1389 ± 0.0023	**0.1396** ± 0.0025
12	0.0528 ± 0.0039 •	0.0542 ± 0.0047 •	0.0501 ± 0.0076 •	0.0542 ± 0.0054 •	0.0538 ± 0.0062 •	0.0550 ± 0.0041 •	**0.0561** ± 0.0000
13	0.0277 ± 0.0024 •	0.0314 ± 0.0056	0.0264 ± 0.0076 •	0.0308 ± 0.0055	0.0310 ± 0.0057	0.0310 ± 0.0055	**0.0321** ± 0.0044
14	0.3209 ± 0.0088 •	**0.3364** ± 0.0121	0.3321 ± 0.0130	0.3360 ± 0.0119	0.3350 ± 0.0128	0.3362 ± 0.0124	0.3362 ± 0.0121
15	**0.0023** ± 0.0007 •	0.0001 ± 0.0004 •	0.0002 ± 0.0006 •	0.0003 ± 0.0010 •	0.0006 ± 0.0011	0.0004 ± 0.0010 •	0.0008 ± 0.0013
16	0.0022 ± 0.0014	0.0017 ± 0.0019	0.0012 ± 0.0013 •	0.0019 ± 0.0019	0.0019 ± 0.0020	0.0019 ± 0.0019	**0.0023** ± 0.0024
17	**0.0026** ± 0.0012 •	0.0014 ± 0.0012 •	0.0009 ± 0.0012 •	0.0013 ± 0.0014 •	0.0017 ± 0.0016	0.0012 ± 0.0012 •	0.0022 ± 0.0018
18	0.4628 ± 0.0060 •	0.4718 ± 0.0077	0.4715 ± 0.0087	**0.4733** ± 0.0072	0.4713 ± 0.0068	0.4718 ± 0.0084	0.4729 ± 0.0079
19	0.2166 ± 0.0097 •	0.2355 ± 0.0123	0.2325 ± 0.0170	0.2317 ± 0.0168	**0.2370** ± 0.0110	0.2309 ± 0.0175	0.2362 ± 0.0111
20	0.1113 ± 0.0096 •	0.1280 ± 0 0127	0.1091 ± 0.0244 •	0.1309 ± 0.0135	0.1314 ± 0.0124	0.1314 ± 0.0117	**0.1320** ± 0.0111
Ave. Rank	5.2000	4.0500	6.2000	3.700	3.0000	4.4000	1.4500

TABLE 6
A Value of Selective Ensemble

	Accuracy						
Data ID	DT	GASEN	RSE	SVMperf-Gmean	Bisection-FM	Gradient-BA	PASE
1	0.8358 ± 0.0113 •	0.8641 ± 0.0195	0.8629 ± 0.0192	0.8637 ± 0.0177	0.8633 ± 0.0191	0.8588 ± 0.0162 •	**0.8694** ± 0.0203
2	0.8259 ± 0.0149 •	0.8586 ± 0.0231	0.8613 ± 0.0261	0.8599 ± 0.0232	**0.8630** ± 0.0267	0.8586 ± 0.0230	0.8624 ± 0.0230
3	0.8648 ± 0.0094 •	0.9425 ± 0.0137 •	0.9452 ± 0.0136 •	0.9416 ± 0.0181 •	0.9495 ± 0.0130	0.9432 ± 0.0148 •	**0.9515** ± 0.0095
4	0.9235 ± 0.0104 •	0.9546 ± 0.0142	0.9549 ± 0.0158	0.9565 ± 0.0162	0.9549 ± 0.0196	0.9570 ± 0.0164	**0.9577** ± 0.0160
5	0.6156 ± 0.0112 •	0.6969 ± 0.0395	0.6892 ± 0.0214 •	0.6866 ± 0.0334 •	0.6914 ± 0.0338 •	0.6899 ± 0.0296	**0.7031** ± 0.0419
6	0.8788 ± 0.0157 •	0.9223 ± 0.0253	0.9196 ± 0.0235	**0.9265** ± 0.0244	0.9208 ± 0.0250	0.9261 ± 0.0243	0.9230 ± 0.0241
7	0.7149 ± 0.0144 •	0.7588 ± 0.0270	0.7550 ± 0.0229	0.7603 ± 0.0238	0.7597 ± 0.0265	0.7583 ± 0.0242	**0.7621** ± 0.0283
8	0.7773 ± 0.0090	**0.7800** ± 0.0098	0.7765 ± 0.0098	0.7792 ± 0.0095	0.7785 ± 0.0102	0.7759 ± 0.0105 •	0.7797 ± 0.0102
9	0.7014 ± 0.0072 •	0.7515 ± 0.0160	0.7468 ± 0.0137	0.7436 ± 0.0157	**0.7520** ± 0.0193	0.7415 ± 0.0138 •	0.7509 ± 0.0199
10	0.7802 ± 0.0098 •	0.8622 ± 0.0201	0.8574 ± 0.0194 •	0.8671 ± 0.0167	**0.8691** ± 0.0176	0.8636 ± 0.0150	0.8678 ± 0.0168
11	0.9892 ± 0.0024 •	0.9932 ± 0.0033	0.9923 ± 0.0042	0.9934 ± 0.0031	0.9938 ± 0.0029	0.9933 ± 0.0030	**0.9939** ± 0.0029
12	0.9924 ± 0.0063 •	0.9951 ± 0.0098 •	0.9869 ± 0.0097 •	0.9951 ± 0.0101	0.9959 ± 0.0067	0.9959 ± 0.0098	**0.9981** ± 0.0038
13	0.9664 ± 0.0057 •	**0.9807** ± 0.0094	0.9798 ± 0.0085	0.9785 ± 0.0111	0.9791 ± 0.0103	0.9776 ± 0.0112	0.9804 ± 0.0091
14	0.9432 ± 0.0085 •	0.9653 ± 0.0132	0.9653 ± 0.0140	0.9639 ± 0.0141	**0.9666** ± 0.0136	0.9639 ± 0.0140	0.9657 ± 0.0126
15	0.9511 ± 0.0029 •	0.9645 ± 0.0026	**0.9658** ± 0.0018 •	0.9642 ± 0.0034	0.9630 ± 0.0037	0.9647 ± 0.0029	0.9642 ± 0.0033
16	0.9729 ± 0.0032 •	0.9789 ± 0.0037	**0.9814** ± 0.0037 •	0.9792 ± 0.0043	0.9788 ± 0.0050	0.9799 ± 0.0040 •	0.9780 ± 0.0047
17	0.9797 ± 0.0017 •	0.9855 ± 0.0022	**0.9858** ± 0.0019	0.9850 ± 0.0020	0.9850 ± 0.0028	0.9848 ± 0.0025	0.9850 ± 0.0034
18	0.9394 ± 0.0080 •	0.9580 ± 0.0118	**0.9600** ± 0.0113	0.9555 ± 0.0106	0.9596 ± 0.0114	0.9540 ± 0.0102 •	0.9591 ± 0.0115
19	0.9458 ± 0.0103 •	0.9887 ± 0.0154	0.9872 ± 0.0170	0.9811 ± 0.0218	0.9842 ± 0.0186	0.9804 ± 0.0221	**0.9887** ± 0.0133
20	0.9381 ± 0.0144 •	0.9760 ± 0.0221 •	0.9606 ± 0.0268 •	0.9777 ± 0.0202	0.9817 ± 0.0208	0.9783 ± 0.0194	**0.9823** ± 0.0199
Ave. Rank	6.8500	3.5000	4.0500	3.950	3.1000	4.4500	2.1000

bisection is F-measure and the objective measure of gradient is the Balanced Accuracy(BA). The trade-off parameter of SVMperf (C in model (52)) is set as 1. The maximal number of iteration of the bisection and the gradient is set as 20.

All of the benchmark methods will output a weight vector. Each weight signifies the important degree of each base classifier. We compare the performance of algorithms in two ensemble settings: selective ensemble and weight ensemble. In the first setting, the classifiers with a weight above the mean value are chosen. The set of the chosen classifiers is

$$H^* = \{h_j | w_j^* > \frac{1}{T}\sum_{k=1}^{T} w_k^*, j = 1, 2, \ldots T\}.$$

Then, the chosen classifiers are combined by majority voting. That is, the final prediction for each instance is positive if $\sum_{h_j \in H^*} h_j(\boldsymbol{x}) > 0$. Otherwise, the prediction is negative. In this setting, we also present the selected number of classifiers of each method. To show the efficiency of the learned weight, we rank the base trees by their weight values and present the vote performance of the ordered trees.

TABLE 7
PA Value of Weight Ensemble

	Pure Accuracy						
Data ID	Bagging	GASEN	RSE	SVMperf-Gmean	Bisection-FM	Gradient-BA	PASE
1	0.7240 ± 0.0332	0.7228 ± 0.0355	0.7251 ± 0.0318	0.7231 ± 0.0344	0.7248 ± 0.0397	0.7252 ± 0.0344	**0.7335** ± 0.0394
2	0.7157 ± 0.0456	0.7140 ± 0.0460	0.7203 ± 0.0457	0.7161 ± 0.0463	0.7230 ± 0.0540	0.7153 ± 0.0452	**0.7268** ± 0.0489
3	0.8858 ± 0.0262 •	0.8792 ± 0.0291 •	0.8881 ± 0.0293 •	0.8814 ± 0.0370 •	0.8902 ± 0.0264 •	0.8858 ± 0.0262 •	**0.9027** ± 0.0232
4	0.9063 ± 0.0359	0.9032 ± 0.0322	0.9077 ± 0.0352	0.9062 ± 0.0352	0.9073 ± 0.0342	0.9063 ± 0.0359	**0.9103** ± 0.0325
5	0.2697 ± 0.0774 •	0.2851 ± 0.0916 •	0.2600 ± 0.0852 •	0.2673 ± 0.0779 •	0.2915 ± 0.0713 •	0.2622 ± 0.0810 •	**0.3303** ± 0.0908
6	0.8356 ± 0.0552	0.8284 ± 0.0564	0.8346 ± 0.0543	0.8348 ± 0.0554	0.8364 ± 0.0584	**0.8364** ± 0.0556	0.8335 ± 0.0564
7	0.4460 ± 0.0536 •	0.4469 ± 0.0550 •	0.4454 ± 0.0562 •	0.4460 ± 0.0538 •	0.4472 ± 0.0558 •	0.4489 ± 0.0541 •	**0.4775** ± 0.0564
8	0.4054 ± 0.0339 •	0.4068 ± 0.0327 •	0.3963 ± 0.0358 •	0.4054 ± 0.0339 •	0.4080 ± 0.0347 •	0.4030 ± 0.0332 •	**0.4315** ± 0.0392
9	0.3124 ± 0.0468 •	0.3180 ± 0.0485 •	0.3140 ± 0.0481 •	0.3122 ± 0.0473 •	0.3307 ± 0.0497 •	0.3079 ± 0.0467 •	**0.3907** ± 0.0390
10	0.6076 ± 0.0600	0.5869 ± 0.0684 •	0.6071 ± 0.0599	0.6083 ± 0.0599	0.6079 ± 0.0594	0.6072 ± 0.0595	**0.6149** ± 0.0571
11	0.9730 ± 0.0127	0.9723 ± 0.0129	0.9733 ± 0.0120	0.9735 ± 0.0128	0.9728 ± 0.0127	0.9730 ± 0.0127	**0.9757** ± 0.0114
12	0.9596 ± 0.0863	0.9558 ± 0.0865	0.9404 ± 0.0858 •	0.9596 ± 0.0863	0.9596 ± 0.0863	0.9596 ± 0.0863	**0.9798** ± 0.0368
13	0.7547 ± 0.0994	0.7482 ± 0.1100	0.7547 ± 0.1047	0.7547 ± 0.0994	0.7547 ± 0.0994	0.7547 ± 0.0994	**0.7624** ± 0.1007
14	0.9226 ± 0.0311	0.9226 ± 0.0310	0.9239 ± 0.0310	0.9226 ± 0.0311	0.9222 ± 0.0319	0.9226 ± 0.0311	**0.9261** ± 0.0287
15	0.0127 ± 0.0393	-0.0001 ± 0.0159	0.0134 ± 0.0401	0.0127 ± 0.0393	0.0127 ± 0.0393	0.0127 ± 0.0393	**0.0292** ± 0.0675
16	0.1218 ± 0.1184	0.1114 ± 0.1202	0.1165 ± 0.1220	0.1211 ± 0.1174	0.1213 ± 0.1183	0.1218 ± 0.1184	**0.1575** ± 0.1321
17	0.1325 ± 0.1256	0.1479 ± 0.1312	0.1264 ± 0.1297	0.1325 ± 0.1256	0.1413 ± 0.1236	0.1325 ± 0.1256	**0.2049** ± 0.1764
18	0.9178 ± 0.0229	0.9167 ± 0.0234	0.9189 ± 0.0227	0.9174 ± 0.0230	0.9189 ± 0.0242	0.9178 ± 0.0229	**0.9218** ± 0.0230
19	0.9770 ± 0.0341	0.9665 ± 0.0452	**0.9791** ± 0.0304	0.9770 ± 0.0341	0.9770 ± 0.0341	0.9770 ± 0.0341	0.9750 ± 0.0341
20	0.9235 ± 0.0848	0.9094 ± 0.0731 •	0.9204 ± 0.0802	0.9235 ± 0.0848	0.9212 ± 0.0851	0.9212 ± 0.0851	**0.9310** ± 0.0750
Ave. Rank	3.6000	5.7500	4.4500	4.4500	3.4500	4.8000	1.5000

TABLE 8
TP Value of Weight Ensemble

	TP						
Data ID	Bagging	GASEN	RSE	SVMperf-Gmean	Bisection-FM	Gradient-BA	PASE
1	0.3982 ± 0.0175	**0.3988** ± 0.0170	0.3978 ± 0.0178	0.3980 ± 0.0183	0.3945 ± 0.0188	**0.3988** ± 0.0186	0.3986 ± 0.0198
2	0.3729 ± 0.0216 •	0.3743 ± 0.0224	0.3735 ± 0.0218 •	0.3733 ± 0.0217 •	0.3770 ± 0.0237	0.3728 ± 0.0212 •	**0.3838** ± 0.0244
3	0.3477 ± 0.0121 •	0.3464 ± 0.0117 •	0.3468 ± 0.0124 •	0.3478 ± 0.0122 •	0.3495 ± 0.0112 •	0.3477 ± 0.0121 •	**0.3548** ± 0.0103
4	0.3530 ± 0.0105	0.3518 ± 0.0110	0.3530 ± 0.0107	0.3527 ± 0.0104	0.3535 ± 0.0099	0.3530 ± 0.0105	**0.3537** ± 0.0090
5	0.1266 ± 0.0305 •	0.1321 ± 0.0290 •	0.1182 ± 0.0328 •	0.1255 ± 0.0304 •	0.1475 ± 0.0387 •	0.1233 ± 0.0324 •	**0.1765** ± 0.0372
6	0.3189 ± 0.0163	0.3185 ± 0.0167	0.3177 ± 0.0165	0.3189 ± 0.0163	0.3189 ± 0.0172	0.3189 ± 0.0163	**0.3196** ± 0.0156
7	0.1983 ± 0.0183 •	0.1991 ± 0.0202 •	0.1970 ± 0.0190 •	0.1983 ± 0.0184 •	0.2024 ± 0.0169 •	0.1990 ± 0.0185 •	**0.2297** ± 0.0154
8	0.1198 ± 0.0194 •	0.1202 ± 0.0188 •	0.1169 ± 0.0197 •	0.1198 ± 0.0194 •	0.1206 ± 0.0197 •	0.1190 ± 0.0196 •	**0.1429** ± 0.0252
9	0.1067 ± 0.0156 •	0.1085 ± 0.0175 •	0.1067 ± 0.0156 •	0.1064 ± 0.0161 •	0.1163 ± 0.0191 •	0.1052 ± 0.0165 •	**0.1676** ± 0.0276
10	0.1463 ± 0.0199	0.1426 ± 0.0207	0.1450 ± 0.0194	0.1468 ± 0.0197	0.1470 ± 0.0208	0.1463 ± 0.0199	**0.1485** ± 0.0192
11	0.1390 ± 0.0022	0.1389 ± 0.0022	0.1390 ± 0.0022	0.1391 ± 0.0022	0.1390 ± 0.0022	0.1390 ± 0.0022	**0.1397** ± 0.0024
12	0.0546 ± 0.0044	0.0542 ± 0.0047 •	0.0527 ± 0.0053 •	0.0546 ± 0.0044	0.0546 ± 0.0044	0.0546 ± 0.0044	**0.0557** ± 0.0019
13	0.0316 ± 0.0051	0.0314 ± 0.0058	0.0310 ± 0.0052	0.0316 ± 0.0051	0.0316 ± 0.0051	0.0316 ± 0.0051	**0.0323** ± 0.0048
14	0.3358 ± 0.0126	0.3360 ± 0.0122	0.3358 ± 0.0123	0.3358 ± 0.0126	0.3358 ± 0.0126	0.3358 ± 0.0126	**0.3378** ± 0.0124
15	0.0003 ± 0.0008	0.0001 ± 0.0004 •	0.0003 ± 0.0008	0.0003 ± 0.0008	0.0003 ± 0.0008	0.0003 ± 0.0008	**0.0008** ± 0.0015
16	0.0018 ± 0.0018	0.0017 ± 0.0019 •	0.0017 ± 0.0019 •	0.0018 ± 0.0018	0.0018 ± 0.0018	0.0018 ± 0.0018	**0.0026** ± 0.0024
17	0.0013 ± 0.0012 •	0.0014 ± 0.0012 •	0.0012 ± 0.0012 •	0.0013 ± 0.0012 •	0.0013 ± 0.0012 •	0.0013 ± 0.0012 •	**0.0022** ± 0.0018
18	0.4724 ± 0.0083	0.4722 ± 0.0072 •	0.4729 ± 0.0078	0.4724 ± 0.0083	0.4724 ± 0.0083	0.4724 ± 0.0083	**0.4747** ± 0.0081
19	0.2377 ± 0.0109	0.2355 ± 0.0135	**0.2385** ± 0.0107	0.2377 ± 0.0109	0.2377 ± 0.0109	0.2377 ± 0.0109	0.2377 ± 0.0109
20	0.1320 ± 0.0132	0.1291 ± 0.0113 •	0.1303 ± 0.0126	0.1320 ± 0.0132	0.1320 ± 0.0132	0.1320 ± 0.0132	**0.1331** ± 0.0115
Ave. Rank	3.2500	5.0000	5.400	4.100	3.550	5.3500	1.3500

In the second setting, all the classifiers are combined by the learned weight. That is, the final prediction for each instance is positive if $h_{\boldsymbol{w}}(\boldsymbol{x}) > 0$, where $h_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{j=1}^{T} w_j h_j(\boldsymbol{x})$. Otherwise, the prediction is negative. The performance of the two settings are compared in terms of PA, TP and A.

Tables 4, 5 and 6 record the comparison results of algorithms in the setting of selective ensemble learning. Each element in the tables denotes the average value ± the standard deviation in terms of the evaluation measure. In each row, the method with the maximum mean value is in bold font; the method is marked a black dot if PASE is significantly better than it in the sense of the pairwise right-tailed Student's test with a confidence level at 90 percent. On each data, the methods are sorted in descending order in terms of the evaluation values. The last row records the average rank of the methods.

From Tables 4, 5, and 6, it is easy to observe that PASE obviously improves the pure accuracy, TP and accuracy values on most data sets. The tables show that PASE obtains the maximal mean values on 14, 13, 9 data sets with respect to pure accuracy, TP and accuracy respectively. According to the last rows in the tables, PASE consistently obtains the first average rank, which indicates that PASE generally outperforms the other methods.

TABLE 9
A Value of Weight Ensemble

	Accuracy						
Data ID	Bagging	GASEN	RSE	SVMperf-Gmean	Bisection-FM	Gradient-BA	PASE
1	0.8624 ± 0.0170	0.8618 ± 0.0182	0.8631 ± 0.0163	0.8620 ± 0.0176	0.8631 ± 0.0200	0.8631 ± 0.0176	**0.8673** ± 0.0198
2	0.8597 ± 0.0223	0.8587 ± 0.0225	0.8620 ± 0.0224	0.8599 ± 0.0227	0.8632 ± 0.0265	0.8595 ± 0.0221	**0.8647** ± 0.0238
3	0.9465 ± 0.0121 •	0.9434 ± 0.0135 •	0.9477 ± 0.0135 •	0.9444 ± 0.0174 •	0.9485 ± 0.0122 •	0.9465 ± 0.0121 •	**0.9543** ± 0.0108
4	0.9560 ± 0.0170	0.9546 ± 0.0152	0.9567 ± 0.0166	0.9560 ± 0.0166	0.9565 ± 0.0162	0.9560 ± 0.0170	**0.9579** ± 0.0155
5	0.6954 ± 0.0286	0.7006 ± 0.0366	0.6947 ± 0.0308 •	0.6947 ± 0.0290	0.6972 ± 0.0248	0.6932 ± 0.0295 •	**0.7046** ± 0.0377
6	0.9246 ± 0.0253	0.9211 ± 0.0259	0.9242 ± 0.0248	0.9242 ± 0.0254	**0.9250** ± 0.0267	**0.9250** ± 0.0255	0.9234 ± 0.0260
7	0.7584 ± 0.0234	0.7586 ± 0.0235	0.7586 ± 0.0244	0.7584 ± 0.0234	0.7572 ± 0.0260	0.7597 ± 0.0237	**0.7626** ± 0.0273
8	0.7794 ± 0.0104	0.7798 ± 0.0105	0.7767 ± 0.0109 •	0.7794 ± 0.0104	**0.7802** ± 0.0106	0.7787 ± 0.0099	**0.7802** ± 0.0092
9	0.7485 ± 0.0165	0.7501 ± 0.0159	0.7493 ± 0.0169	0.7487 ± 0.0164	**0.7509** ± 0.0160	0.7475 ± 0.0163	0.7472 ± 0.0210
10	0.8678 ± 0.0171	0.8610 ± 0.0200 •	0.8682 ± 0.0170	0.8678 ± 0.0171	0.8677 ± 0.0169	0.8677 ± 0.0169	**0.8698** ± 0.0162
11	0.9934 ± 0.0031	0.9932 ± 0.0032	0.9935 ± 0.0029	0.9935 ± 0.0031	0.9934 ± 0.0031	0.9934 ± 0.0031	**0.9941** ± 0.0028
12	0.9955 ± 0.0098	0.9951 ± 0.0098	0.9936 ± 0.0096 •	0.9955 ± 0.0098	0.9955 ± 0.0098	0.9955 ± 0.0098	**0.9978** ± 0.0041
13	0.9804 ± 0.0087	0.9800 ± 0.0096	**0.9809** ± 0.0088	0.9804 ± 0.0087	0.9804 ± 0.0087	0.9804 ± 0.0087	0.9807 ± 0.0094
14	0.9647 ± 0.0140	0.9647 ± 0.0140	0.9653 ± 0.0140	0.9647 ± 0.0140	0.9645 ± 0.0144	0.9647 ± 0.0140	**0.9662** ± 0.0129
15	0.9646 ± 0.0029	0.9646 ± 0.0026	**0.9648** ± 0.0031 •	0.9646 ± 0.0029	0.9646 ± 0.0029	0.9646 ± 0.0029	0.9638 ± 0.0031
16	**0.9795** ± 0.0039 •	0.9787 ± 0.0039	**0.9795** ± 0.0041 •	0.9795 ± 0.0041 •	0.9795 ± 0.0040 •	**0.9795** ± 0.0039 •	0.9771 ± 0.0051
17	0.9852 ± 0.0023	**0.9854** ± 0.0024	0.9853 ± 0.0023	0.9852 ± 0.0023	0.9853 ± 0.0023	0.9852 ± 0.0023	0.9848 ± 0.0035
18	0.9589 ± 0.0115	0.9584 ± 0.0117	0.9595 ± 0.0114	0.9587 ± 0.0115	0.9595 ± 0.0121	0.9589 ± 0.0115	**0.9609** ± 0.0115
19	0.9917 ± 0.0123	0.9879 ± 0.0162	**0.9925** ± 0.0109	0.9917 ± 0.0123	0.9917 ± 0.0123	0.9917 ± 0.0123	0.9909 ± 0.0123
20	0.9811 ± 0.0221	0.9777 ± 0.0198 •	0.9806 ± 0.0210	0.9811 ± 0.0221	0.9806 ± 0.0222	0.9806 ± 0.0222	**0.9829** ± 0.0202
Ave. Rank	3.4500	5.3500	3.1500	4.500	3.6500	5.100	2.8000

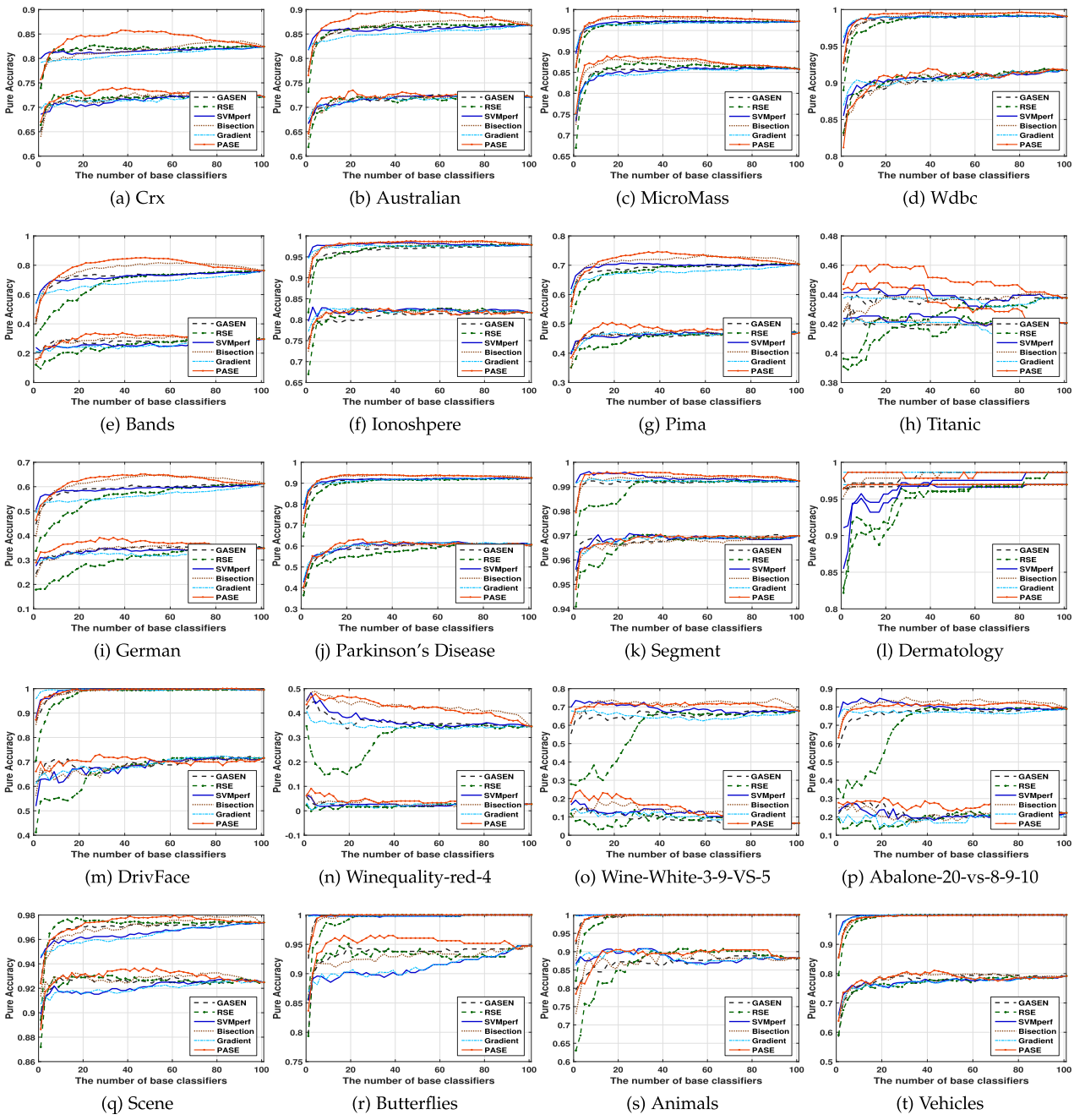
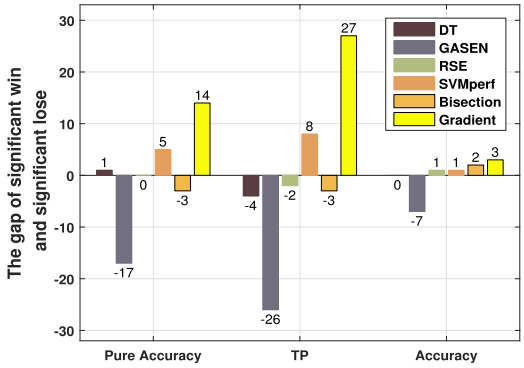


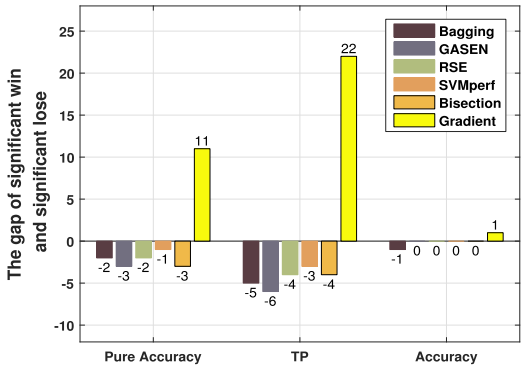
Fig. 5. The average train and test pure accuracy curves of ordered ensemble with the increasing of the ensemble number.

Tables 7, 8 and 9 record the comparison results of algorithms in the setting of weight ensemble learning. The elements in the tables are the same as the ones in Tables 4, 5, and 6. From Tables 7 and 8, we observe that PASE obtains the highest PA and TP value in most cases. From Table 9, we observe that PASE obtains a comparable A value with RSE, while the TP value of RSE is lower. The results signify that PA is a more balanced performance measure, and both the weight learned by optimizing PA and the algorithm that optimizes PA are efficient. We also show the number of selected trees in Table S1 in the supplementary material, available online.

In optimization based selective ensemble learning, it is a traditional method to set the mean weight as the threshold for choosing classifiers. In this paper, we follow this traditional method. To give more insight into the weight learned by different methods and the threshold used in truncating the weights, we present the performance of bagging with ordered trees. That is, for each method, the base trees are ranked according to the value of weight, and then used to ensemble. Concretely, for T trees $\{h_1, h_2, \ldots, h_T\}$, suppose their weight order is $w^*_{r_1} > w^*_{r_2} > \ldots > w^*_{r_T}$, then the tree is ranked as $h_{r_1}, h_{r_2}, \ldots, h_{r_T}$, where $r_i \in \{1, 2, \ldots, T\}$. Then an odd number of ordered trees are used to ensemble by voting. Fig. 5 shows the average train and test pure accuracy curves of ordered ensemble with the increasing of the ensemble number. The points on the curve present the performance of combining $1, 3, .., T$ ordered trees in turn. On each data, the top curves are the results on the



(a) The Result of Selective Ensemble



(b) The Result of Weight Ensemble

Fig. 6. Statistical Comparison Results.

training data sets and the down ones are on the test data sets. From Fig. 5, it is easy to observe that in most cases, PASE obtains the best performance if the selective number is the same. This signifies that compared with the traditional algorithms, the weight learned by PASE is effective in selecting classifiers and the thresholds for selecting the good set of classifiers are broad.

To further analyze the results, we compare the difference between the significant better and the significance worse at the significance level of 95% [64]. The bars of Fig. 6 depict the results. From Fig. 6, we observe that PASE obtains the largest gap, which indicates that PASE is significantly better than the other methods. Please refer to the supplementary material, available online, for more details about how to depict the bars of Fig. 6.

4.4 Experimental Comparison With Boosting Algorithms

In ensemble learning, the strength and the diversity of the base classifiers are two key factors for the ensemble performance. For bagging, the strength of the base classifiers is strong while their diversity is small. However, for boosting, the base classifiers are complementary while their performance only needs to be slightly better than a weak classifier (the classifier with performance comparable with random guess).

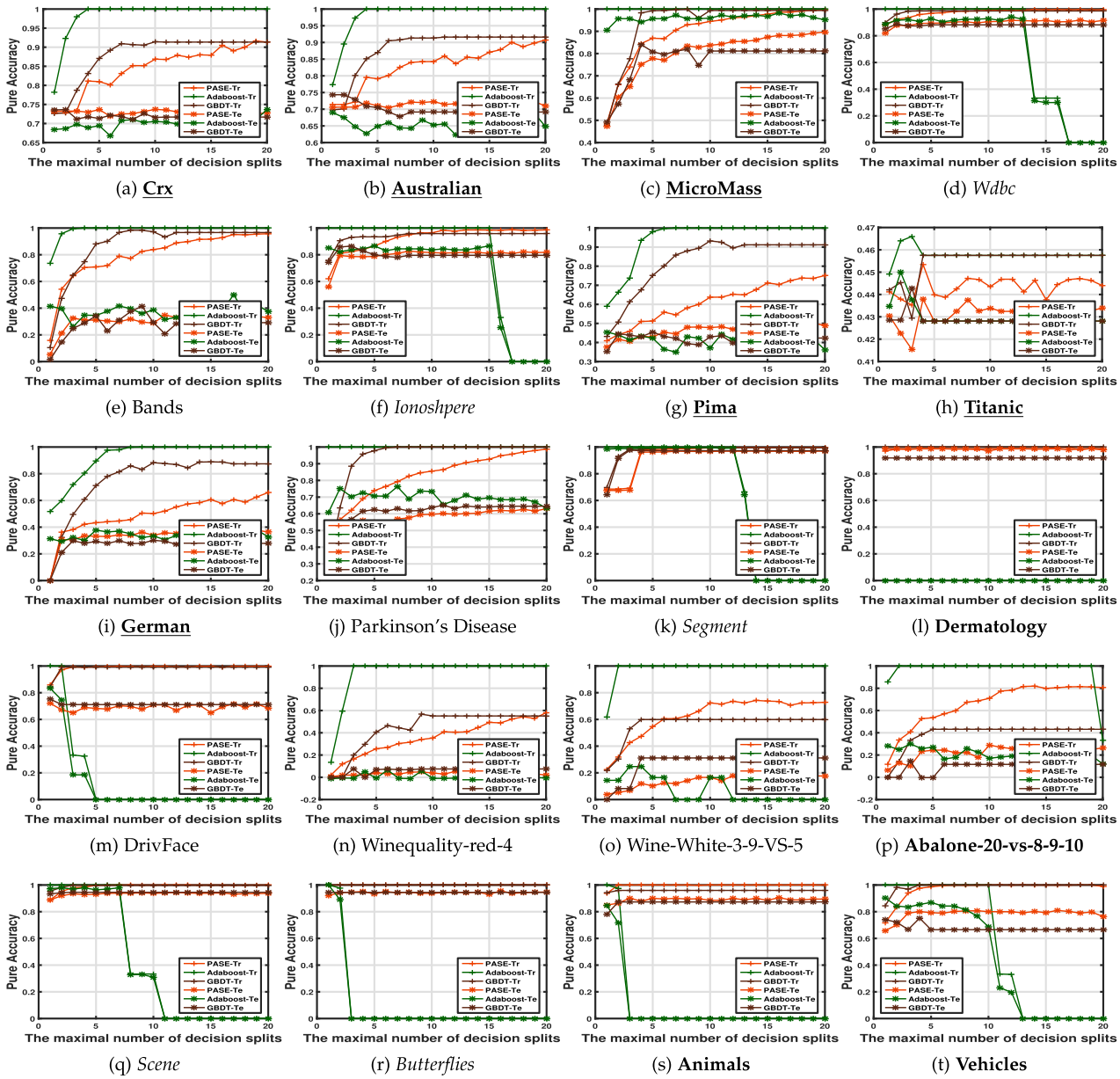
As is well known, a weak classifier is proven to be equivalent to a strong classifier through boosting by the boosting algorithms. In this subsection, through the image data sets, we investigate which type of base classifiers that PASE applies to, and compare PASE with AdaboostM1 and Gradient Boosting Decision Tree (GBDT). The AdaboostM1 is implemented by the Fitensemble package in MATLAB. The GBDT is implemented by XGboost and the objective function of GBDT is AUC.

The strength of the base tree can be controlled by the maximal number of decision splits. The more the decision splits is, the deeper the tree is and the stronger the performance is. We compare the performance of algorithms under different maximal number of decision splits. The experimental data sets are still the twenty data sets in Table 3. Each data set is randomly divided into a training set and a test set at a ratio of 7:3. On each division, we run every method 12 times to evaluate the average performance. The performance is evaluated in terms of pure accuracy, as shown in the Fig. 7. We also respectively show the associated results w.r.t TP and accuracy in Figs. S3 and S4 in the supplementary material, available online.

From Fig. 7, we observe that Adaboost applies to boosting weak classifiers on 9/20 data sets (Wdbc, Ionoshpere, Segment, DrivFace, Wine-White-3-9-VS-5, Scene, Butterflies, Animals, Vehicles), while PASE and GBDT apply to boosting classifier of various strengths. In addition, we observe that PASE obtains the highest test PA value on most of the split numbers on 10/20 data sets. The names of these data sets in the subgraphs are in bold. Further, PASE obtains the lowest train PA value on most of the split numbers on 6/10 of the above bold data sets. The names of these data sets in the subgraphs are underlined. On 5/20 data sets, whose names in the subgraphs are italic, PASE obtains comparable test PA value with GBDT and higher values than Adaboost in many cases. Above all, we can conclude that PASE is not easy to overfit and performs better than the two boosting algorithms in most cases.

5 Conclusion and Future Work

In this paper, we worked on the pure accuracy measure, which eliminates random consistency from the accuracy measure. We illustrated that the pure accuracy measure is more class distribution insensitive and more discriminative than accuracy and F-measure. We proposed a tighter concentration inequality and then developed a generalization bound on the pure accuracy measure, which is tighter than the existing bound. After that, we designed a learning algorithm optimizing the pure accuracy measure and used it into the selective ensemble learning. Experimental results on twenty data sets indicated that PASE outperforms the other eight representative learning algorithms, including AdaboostM1 and GBDT. It is worth mentioning that GBDT could be extended to optimize the PA, by means of a gradient approach, which would be very interesting to study in the future. Besides, we are interested in axiomatically defining the pure consistency measure in other popular learning tasks, such as, imbalanced learning, multi-class classification, multi-label learning and deep learning.



(a) **Crx** (b) **Australian** (c) **MicroMass** (d) *Wdbc*

(e) Bands (f) *Ionoshpere* (g) **Pima** (h) **Titanic**

(i) **German** (j) Parkinson's Disease (k) *Segment* (l) **Dermatology**

(m) DrivFace (n) Winequality-red-4 (o) Wine-White-3-9-VS-5 (p) **Abalone-20-vs-8-9-10**

(q) *Scene* (r) *Butterflies* (s) **Animals** (t) **Vehicles**

Fig. 7. The average train and test pure accuracy curves with the increasing of the maximal number of decision splits.

REFERENCES

[1] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[2] J. Wang, Y. Qian, and F. Li, "Learning with mitigating random consistency from the accuracy measure," *Mach. Learn.*, vol. 109, pp. 2247–2281, 2020.

[3] J. Wang, Y. Qian, F. Li, and G. Liu, "Support vector machine with eliminating the random consistency," *J. Comput. Res. Develop.*, vol. 57, no. 8, pp. 1581–1593, 2020.

[4] D. L. Sabers and L. S. Feldt, "An empirical study of the effect of the correction for chance success on the reliability and validity of an aptitude test," *J. Educ. Meas.*, vol. 5, no. 3, pp. 251–258, 1968.

[5] J. Diamond and W. Evans, "The correction for guessing," *Rev. Educ. Res.*, vol. 43, no. 2, pp. 181–191, 1973.

[6] D. Budescu and M. Bar-Hillel, "To guess or not to guess: A decision-theoretic view of formula scoring," *J. Educ. Meas.*, vol. 30, no. 4, pp. 277–291, 1993.

[7] M. P. Espinosa and J. Gardeazabal, "Optimal correction for guessing in multiple-choice tests," *J. Math. Psychol.*, vol. 54, no. 5, pp. 415–425, 2010.

[8] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[9] W. A. Scott, "Reliability of content analysis: The case of nominal scale coding," *Public Opin. Quart.*, vol. 19, no. 3, pp. 321–325, 1955.

[10] L. A. Goodman and W. H. Kruskal, "Measures of association for cross classifications," *Pub. Amer. Statist. Assoc.*, vol. 49, no. 268, pp. 732–764, 1963.

[11] A. N. Albatineh, M. Niewiadomskabugaj, and D. Mihalko, "On similarity indices and correction for chance agreement," *J. Classification*, vol. 23, no. 2, pp. 301–313, 2006.

[12] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[13] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Is a correction for chance necessary?," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 1073–1080.

[14] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, 2010.

[15] A. J. Gates and Y. Ahn, "The impact of random models on clustering similarity," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 3049–3076, 2017.

[16] P. L. Bartlett, M. I. Jordan, and J. Mcauliffe, "Convexity, classification, and risk bounds," *J. Amer. Statist. Assoc.*, vol. 101, no. 473, pp. 138–156, 2006.

[17] T. Zhang, "Statistical behavior and consistency of classification methods based on convex risk minimization," *Ann. Statist.*, vol. 32, no. 1, pp. 56–134, 2003.
[18] Z. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artif. Intell.*, vol. 137, pp. 239–263, 2002.
[19] G. Martinezmunoz and A. Suarez, "Pruning in ordered bagging ensembles," in *Proc. Int. Conf. Mach. Learn.*, 2006, pp. 609–616.
[20] L. G. Valiant, "A theory of the learnable," *Commun. ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.
[21] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, USA: Springer, 1999.
[22] P. L. Bartlett and S. Mendelson, "Rademacher and Gaussian complexities: Risk bounds and structural results," *J. Mach. Learn. Res.*, vol. 3, no. 3, pp. 463–482, 2003.
[23] M. Ledoux and M. Talagrand, "Probability in banach spaces: Isoperimetry and processes," *Ergebnisse Der Mathematik Und Ihrer Grenzgebiete*, vol. 23, pp. 1–186, 2002.
[24] R. E. Schapire, Y. Freund, P. L. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Ann. Statist.*, vol. 26, no. 5, pp. 1651–1686, 1998.
[25] M. Zhao, N. U. Edakunni, A. C. Pocock, and G. Brown, "Beyond fano's inequality: Bounds on the optimal f-score, BER, and cost-sensitive risk and their implications," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 1033–1090, 2013.
[26] W. Waegeman, K. Dembczyński, A. Jachnik, W. Cheng, and E. Hüllermeier, "On the bayes-optimality of f-measure maximizers," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3333–3388, 2014.
[27] A. Sanyal, P. Kumar, P. Kar, S. Chawla, and F. Sebastiani, "Optimizing non-decomposable measures with deep networks," *Mach. Learn.*, vol. 107, pp. 1597–1620, 2018.
[28] O. O. Koyejo, N. Natarajan, P. K. Ravikumar, and I. S. Dhillon, "Consistent binary classification with generalized performance metrics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2744–2752.
[29] W. Kotlowski and K. Dembczyński, "Surrogate regret bounds for generalized classification performance metrics," in *Proc. Asian Conf. Mach. Learn.*, 2016, pp. 301–316.
[30] A. Menon, H. Narasimhan, S. Agarwal, and S. Chawla, "On the statistical consistency of algorithms for binary classification under class imbalance," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 603–611.
[31] H. Narasimhan, H. G. Ramaswamy, A. Saha, and S. Agarwal, "Consistent multiclass algorithms for complex performance measures," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2398–2407.
[32] K. Dembczyński, W. Kotłowski, O. Koyejo, and N. Natarajan, "Consistency analysis for binary classification revisited," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 961–969.
[33] H. Narasimhan, R. Vaish, and S. Agarwal, "On the statistical consistency of plug-in classifiers for non-decomposable performance measures," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1493–1501.
[34] S. A. P. Parambath, N. Usunier, and Y. Grandvalet, "Optimizing f-measures by cost-sensitive classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2123–2131.
[35] H. Narasimhan, "Learning with complex loss functions and constraints," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 1646–1654.
[36] H. Narasimhan, P. Kar, and P. Jain, "Optimizing non-decomposable performance measures: A tale of two classes," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 199–208.
[37] T. Joachims, "A support vector method for multivariate performance measures," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 377–384.
[38] T. Hazan, J. Keshet, and D. Mcallester, "Direct loss minimization for structured prediction," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2010, pp. 1594–1602.
[39] Y. Song, A. G. Schwing, R. S. Zemel, and R. Urtasun, "Training deep neural networks via direct loss minimization," *Comput. Sci.*, vol. 48, pp. 2169–2177, 2015.
[40] S. M. Huan Xu, "Robustness and generalization," *Mach. Learn.*, vol. 86, pp. 391–423, 2012.
[41] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *Proc. Int. Conf. Algorithmic Learn. Theory*, 2005, pp. 63–77.
[42] J. Mooij, D. Janzing, J. Peters, and B. Schölkopf, "Regression by dependence minimization and its application to causal inference in additive noise models," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 745–752.
[43] F. Yu, D. Liu, S. Kumar, J. Tony, and S.-F. Chang, "$\propto$SVM for learning with label proportions," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 504–512.
[44] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[45] A. Caliskan, J. J. Bryson, and A. Narayanan, "Semantics derived automatically from language corpora contain human-like biases," *Science*, vol. 356, no. 6334, pp. 183–186, 2017.
[46] O. Bousquet, "A bennett concentration inequality and its application to suprema of empirical processes," *Comptes Rendus Mathematique*, vol. 334, no. 6, pp. 495–500, 2002.
[47] S. Agarwal, T. Graepel, R. Herbrich, S. Harpeled, and D. Roth, "Generalization bounds for the area under the ROC curve," *J. Mach. Learn. Res.*, vol. 6, no. 2, pp. 393–425, 2005.
[48] C. Cortes, M. Mohri, and U. Syed, "Deep boosting," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1179–1187.
[49] W. Gao and Z. H. Zhou, "On the doubt about margin explanation of boosting," *Artif. Intell.*, vol. 203, no. 5, pp. 1–18, 2013.
[50] L. Zhang, T. Yang, and R. Jin, "Empirical risk minimization for stochastic convex optimization: $O(1/n)$- and $O(1/n^2)$-type of risk bounds," in *Proc. 30th Conf. Learn. Theory*, 2017, pp. 1954–1979.
[51] S. Boucheron, G. Lugosi, and P. Massart, *Concentration Inequalities: A Nonasymptotic Theory of Independence*. London, U.K.: Oxford Univ. Press, 2013.
[52] H. U. S. Ziyuan Gao, C. Ries and S. Zilles, "Preference-based teaching," *J. Mach. Learn. Res.*, vol. 18, pp. 31:1–31:32, 2017.
[53] E. Alanazi, M. Mouhoub, and S. Zilles, "The complexity of exact learning of acyclic conditional preference networks from swap examples," *Artif. Intell.*, vol. 278, 2020, Art. no. 103182.
[54] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2012.
[55] S. Boucheron, G. Lugosi, and P. Massart, "A sharp concentration inequality with applications," *Random Struct. Algorithms*, vol. 16, no. 3, pp. 277–292, 2000.
[56] N. Li and Z. Zhou, "Selective ensemble under regularization framework," in *Proc. Int. Workshop Mult. Classifier Syst.*, 2009, pp. 293–303.
[57] T. Ibaraki, "Parametric approaches to fractional programs," *Math. Program.*, vol. 26, no. 3, pp. 345–362, 1983.
[58] H. Wang, F. Nie, and H. Huang, "Robust distance metric learning via simultaneous L1-norm minimization and maximization," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1836–1844.
[59] R. W. Freund and F. Jarre, "Solving the sum-of-ratios problem by an interior-point method," *J. Glob. Optim.*, vol. 19, no. 1, pp. 83–102, 2001.
[60] Y. D. Sergeyev, "Global one-dimensional optimization using smooth auxiliary functions," *Math. Program.*, vol. 81, no. 1, pp. 127–146, 1998.
[61] D. Dua and C. Graff, UCI machine learning repository, 2019. School Inform. Comput. Sci., Univ. California, Irvine, CA, USA, 2019. [Online]. Available: http://archive.ics.uci.edu/ml/datasets
[62] J. Alcalafdez *et al.*, "Keel: A software tool to assess evolutionary algorithms for data mining problems," *Soft Comput.*, vol. 13, no. 3, pp. 307–318, 2008.
[63] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[64] F. Li, Y. Qian, J. Wang, and J. Liang, "Multigranulation information fusion: A dempster-shafer evidence theory-based clustering ensemble method," *Inf. Sci.*, vol. 378, pp. 389–409, 2017.

Jieting wang (Member, IEEE) received the MS and PhD degrees in computers with applications from Shanxi University, Taiyuan, China, in 2011 and 2021, respectively. She is currently a teacher with the Institute of Big Data Science and Industry, Shanxi University. Her research interest includes statistical machine learning and ensemble learning.

Yuhua Qian (Member, IEEE) received the MS and PhD degrees in computers with applications from Shanxi University, Taiyuan, China, in 2005 and 2011, respectively. He is currently a director with the Institute of Big Data Science and Industry, Shanxi University, where he is also a professor with the Key Laboratory of Computational Intelligence and Chinese Information Processing, Ministry of Education. He is best known for artificial intelligence, machine learning and machine vision. He has authored more than 100 articles on these topics in international journals.

Feijiang Li (Member, IEEE) received the PhD degree in computers with applications from Shanxi University, Taiyuan, China, in 2020. He is currently a teacher with the Institute of Big Data Science and Industry, Shanxi University. His research interest includes machine learning and knowledge discovery.

Jiye Liang (Senior Member, IEEE) received the MS and PhD degrees from Xi'an Jiaotong University, Xi'an, China, in 1990 and 2001, respectively. He is currently a professor with the School of Computer and Information Technology, Shanxi University, Taiyuan, China, where he is also the director of the Key Laboratory of Computational Intelligence and Chinese Information Processing of the Ministry of Education. He has authored more than 170 journal papers in his research fields. His current research interests include computational intelligence, granular computing, data mining, and knowledge discovery.

Qingfu Zhang (Fellow, IEEE) received the BSc degree in mathematics from Shanxi University, China in 1984, the MSc degree in applied mathematics and the PhD degree in information engineering from Xidian University, China, in 1991 and 1994, respectively. He is the chair professor of computational intelligence with the Department of Computer Science, City University of Hong Kong. His main research interests include evolutionary computation, optimization, neural networks, machine learning, and their applications. He is an associate editor for the *IEEE Transactions on Evolutionary Computation* and *IEEE Transactions on Cybernetics*. He is a Web of Science highly cited researcher in computer science.