International Journal of Automation and Computing 17(5), October 2020, 621-636
DOI: 10.1007/s11633-020-1239-y

Item Response Theory Based Ensemble in Machine Learning

Ziheng Chen Hongshik Ahn

Department of Applied Mathematics and Statistics, Stony Brook University, New York 11794−3600, USA

Abstract: In this article, we propose a novel probabilistic framework to improve the accuracy of a weighted majority voting algorithm. In order to assign higher weights to the classifiers which can correctly classify hard-to-classify instances, we introduce the item response theory (IRT) framework to evaluate the samples′ difficulty and classifiers′ ability simultaneously. We assigned the weights to classifiers based on their abilities. Three models are created with different assumptions suitable for different cases. When making an inference, we keep a balance between the accuracy and complexity. In our experiment, all the base models are constructed by single trees via bootstrap. To explain the models, we illustrate how the IRT ensemble model constructs the classifying boundary. We also compare their performance with other widely used methods and show that our model performs well on 19 datasets.

Keywords: Classification, ensemble learning, item response theory, machine learning, expectation maximization (EM) algorithm.

1 Introduction

Classification ensembles are increasingly gaining attention from the area of machine learning, especially when we focus on improving the accuracy. The most important feature distinguishing the ensemble learning from other types of learning is that it combines the predictions from a group of classifiers rather than depending on a single classifier[1]. It is proved in many cases that the aggregated performance metrics, such as bagging, boosting and incremental learning outperform others without a collective decision strategy.

If one had to identify an idea as central and novel to ensemble learning, it is the combination rule, which can be characterized in two ways: simple majority voting and weighted majority voting. Simple majority voting is just a decision rule which combines the decisions of the classifiers in the ensemble[1]. It is widely applied in ensemble learning due to its simplicity and applicability[2]. Weighted majority voting can be done by multiplying a weight to the decision of each classifier to reflect its ability, and then make the final decision by combining the weighted decisions[3]. These two methods utilize the ability of classifiers based on their performance on training the data. Thus it does not require any parameter tuning once the individual classifiers have been trained.

Here we propose a novel probabilistic framework for the weighted voting classification ensemble. We treat each data point as a problem and different classifier as a subject taking an exam in class. As we know, the performance of one student on a problem depends on two major factors: difficulty of the problem and competence of the student[4]. In the training data, some have significant features and are easy to classify, whereas some are difficult cases because they are near class boundaries. Thus, similar to an exam in class, we define the competence of a classifier as the capability of correctly classifying difficult cases, rather than the number of correctly classified cases. For instance, suppose a classifier correctly classifies some easy cases but fails to deal with difficult cases. Another classifier correctly classifies some difficult cases, while incorrectly classifies easy cases. Then it makes sense that a higher weight is given to the second classifier than the first one.

In this paper, we propose a method which can simultaneously evaluate the ability of a classifier and difficulty of classifying a case. Here, we employ the item response theory (IRT) framework[5, 6], which is widely applied to psychological and educational research, to estimate the latent ability of classifiers.

2 Motivation and background

2.1 Classifier′s ability

Classifier i's ability is defined by the parameter θ_i, which measures its capacity to handle different samples. Not only the number of cases it can classify, but difficulty of each case is also considered when estimating the parameter. The classifier's ability is directly connected with the weight assigned to each classifier in the en-

Research Article
Manuscript received February 16, 2020; accepted June 4, 2020; published online September 9, 2020
Recommended by Associate Editor Matjaz Gams

semble, which is a real value between 0 and 1. A classifier having highly negative ability leads to a weight close to 0, while the opposite is true for the classifier with highly positive ability. Outliers, observations near the boundary and observations surrounded by multiple observations from the other class can usually only be correctly classified by a classifier with a high value of the ability.

2.2 Item response theory

Item response theory considers a set of models that relate responses given to items to latent abilities of the respondents[5]. It is widely applied in educational testing and psychological evaluation. In this model, the probability of a response is a function of the classifier's ability and observation's difficulty. As one classifier either correctly classifies or incorrectly classifies an observation in our case, we only focus on the dichotomous models. In the original model, their relationship can be described as

$$P(y_{ij}=1|\theta_i,\beta_j)=\Phi(\theta_i-\beta_j)=\int_{-\infty}^{\theta_i-\beta_j}\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{t^2}{2}}\,\mathrm{d}t$$

where y_{ij} is a binary response of a classifier i to observation j, $y_{ij}=1$ for a correct classification and $y_{ij}=0$ otherwise. The parameter θ_i is latent ability parameter for classifier i and β_j is the difficulty of observation j. As it only contains one item parameter β_j, it is named 1PNO. For the 2PNO model and 3PNO model, we will introduce 2 item parameters and 3 item parameters correspondingly. In our first method, we will use the 3PNO as the basic framework. In our second method, we design a 2PNO.

2.3 Related works

Many prior works focus on assigning the weights to different classifiers constructed by bootstrapping[7]. A hybrid voting method is proposed in [8], combining the results from different classifiers. Rojarath et al.[9] suggest a method to improve each classifier's ability based on the result of others. A ranking measurement to evaluate the instance's preference toward the classifiers is defined in [10]. Although fast and efficient, all these methods fail to consider the classifiers' abilities. The problem of having different distribution between training data and testing data is tackled via transfer learning[11]. However, they need a powerful pre-train model. The weighted ensemble is applied into high-dimensional cases in [12] and [13]. Ensemble pruning algorithms are also considered in [14] and [15], focusing on pruning the ensembles with significant features.

Recently, probabilistic models are also adopted in ensemble learning. For a comprehensive weight majority vote, recall the naive Bayes combiner is presented in [16]. Based on their probabilistic framework, four ensemble methods are derived subsequently by progressively relaxing the assumption. They construct the model directly by considering the misclassification probability, rather than the difficulty of the observations and the ability of the classifier. In [17], although instance difficulty is taken into consideration, they didn't discuss the detailed decision mechanism between the classifiers and the observations. Weights are assigned to the class distribution at leaf nodes, which requires a large number of parameters as the feature dimension increases[18].

As a fundamental method to improve the classification performance, ensemble learning is used in different tasks. The text mining methods are combined with majority voting to detect the fake news on websites[19]. A double layer Bayesian neural network is designed and applied into multiple data sets[20]. Bagging, boosting and stacking methods are used to assess the credit score of applicants[21].

We propose the IRT ensemble to evaluate the difficulty of samples and the ability of classifiers simultaneously. To the best of our knowledge, our method is the first approach introducing the IRT into the ensemble learning and giving a statistical explanation of the samples' difficulty and classifiers' ability. This work is similar to WAVE[4]. However, although they use a similar idea, they didn't statistically explain the weight. Our proposed methods model the weight assignment in a probabilistic way and can explain the corresponding relationship between the classifier's ability and observations' difficulty.

3 Model development

In this section, we introduce the proposed method in detail. We treat the classifiers as competitors and want to evaluate their performance by considering the accuracy in classifying hard-to-classify samples. The basic rule is to assign a higher weight to the classifiers with higher accuracy in classification. Thus, we adopt the framework in the item response theory[22], which is widely applied in the educational testing to evaluate the items and people simultaneously[23, 24]. We firstly describe our model and explain why it works. Then we introduce two methods to make an inference. Throughout the paper, vectors or matrices are denoted using boldface.

3.1 Model description

We consider a set of classifiers $\Omega=\{C_1,C_2,\cdots,C_n\}$ and a set of data points $\Phi=\{S_1,S_2,\cdots,S_m\}$. For each classifier C_i, there is a corresponding parameter θ_i to denote the ability of the classifier. Similarly, we assign a difficulty parameter β_j for each data point S_j. Based on the IRT framework, the probability of a response for a data point is a function of the classifier's ability and difficulty of classifying the case. Although there have been various models developed within the IRT framework, we focus on the basic unidimensional IRT model because it models the classifier-sample interaction by two single unified traits θ and β. In this model, the response generated from the interaction has only two choices: success or failure. In our problem, success means the classifier recognizing the label of a sample correctly.

Now we can formulate the model. Suppose we have k classifiers and n data points in a training set. We denote the $k \times n$ matrix Y as the performance matrix. For each element Y_{ij}, 1 is assigned if classifier i correctly classifies sample j, or 0 is assigned otherwise. We also define l as an $n \times 1$ vector with all vector elements being 1. Finally, I is a $k \times k$ identity matrix. For an easy description, we first propose Assumptions 1 and 2.

Assumption 1. The performance parameter of a classifier and difficulty parameter of a sample are rational numbers, and the probability of a correct classification can be expressed as a cumulative distribution function (CDF).

Assumption 2. The classifiers give their decisions independently conditioned upon the training data.

For each classifier C_i, the probability of a correct classification for sample point S_j equals to:

$$P(Y_{ij} = 1) = \phi(\alpha_j\theta_i - \beta_j) + \gamma_j(1 - \phi(\alpha_j\theta_i - \beta_j))$$

where $\phi(x)$ is a CDF. Now we explain extra parameters and their original purpose.

1) The discrimination parameter α_j of case S_j reflects the steepness of the probability function. If we set $\gamma_j = 0$ and differentiate the function ϕ of θ_i, then the derivative is $\alpha_j\phi'(\alpha_j\theta_i - \beta_j)$, and α_j serves as the multiplier. The larger the value of α_j is, the steeper the probability is. Hence at some point, any small improvement of the classifier's ability can make a huge difference in the response, which can be used to detect subtle differences in the ability of the classifier.

2) We also define γ_j as a guessing parameter. There is a small probability that a classifier can correctly classify the situation without really learning the features from the training data.

We can see the advantage of the model. The performance of a classifier is estimated based on the responses to discriminating items with different levels of difficulty, but not by the accuracy. Classifiers which correctly classify the difficult cases will get a high estimated value of the performance parameter. Hard-to-classify data points tend to be correctly classified by highly performing classifiers.

3.2 Inference

According to the second assumption, we can write the likelihood function as

$$f(y|\alpha,\gamma) = \prod_{i=1}^{n}\prod_{j=1}^{k} P(Y_{ij} = 1)^{Y_{ij}}(1 - P(Y_{ij} = 1))^{1-Y_{ij}}.$$

If we directly optimize the likelihood function, it will be very unstable due to the non-convexity of the function. Thus, we consider using Markov Chain Monte Carlo (MCMC) to estimate the parameters[25, 26]. The basic idea of MCMC is to use a series of Markov chains and transition kernel to estimate the parameters. After the procedure, we obtain an estimation of the parameters. To make a concise notation, we denote a set of all the parameters as Θ, which includes all the discrimination parameters α_j, the guessing parameters γ_j, difficulty parameters β_j and performance parameters θ_i. For the training data, we denote the sample space as Π. Thus the joint probability function can be written as $f(\Pi|\Theta)$.

1) Metropolis hastings algorithm. The metropolis hastings (M-H) algorithm is a very basic method in the MCMC family, which constructs the Markov chain from the parameter's posterior distribution with a proposal kernel[27]. If we supply a prior distribution $f(\Theta)$ to the parameters, we can write the joint distribution of the parameters and data as $f(\Pi,\Theta) = f(\Pi|\Theta)f(\Theta)$, and following the Bayes rule, the posterior distribution of Θ is

$$p(\Theta|\Pi) = \frac{f(\Pi|\Theta)f(\Theta)}{\int f(\Pi|\Theta)f(\Theta)\mathrm{d}\Theta} \propto f(\Pi|\Theta)f(\Theta).$$

Thus, we get the posterior distribution of certain parameter Θ.

Then we grow the Markov chain by sampling from the posterior distribution in multiple steps[28]. In particular step i of the algorithm, in which the target parameter is Θ^i, we draw a sample Θ' from the proposal kernel $q(\Theta'|\Theta^i)$. The probability of accepting it as the value of Θ^{i+1} in next iteration is

$$\alpha = \min\left\{\frac{f(\Theta^{'}|\text{rest})q(\Theta^{'}|\Theta^i)}{f(\Theta^i|\text{rest})q(\Theta^i|\Theta')}, 1\right\}.$$

Here, the rest parameters are denoted as rest. Suppose $u \leq \alpha$, then we can generate another random number u from uniform distribution U(0,1). We update Θ^{i+1} as Θ'. Otherwise, we reject the proposal value.

The above is a brief summary of the M-H algorithm. In practice, we have to estimate an array of parameters rather than a single parameter, so we need to decompose the parameter vector into different components and update them one by one through the M-H algorithm. This slows down the M-H algorithm. Besides, the prior distribution of the parameters should be carefully determined by considering the effectiveness of the rejection process. Last but not least, the initial value also plays a fundamental role in the efficiency of estimating the parameters.

In this model, we assign the parameters with the following priors:

$$\begin{aligned}&\theta_i \sim \mathrm{N}(0, \sigma_i^2)\\ &\log(\alpha_j) \sim \mathrm{N}(\mu_a, \sigma_a^2) \ \ (\text{we constrain } \alpha_j \geq 0)\\ &\beta_j \sim \mathrm{N}(0, \sigma^2)\\ &\sigma_i^2 \sim \mathrm{IG}(\alpha_\theta, \beta_\theta).\end{aligned}$$

Then we derive the posterior probability density function based on the given prior. We can use Pystan to construct the model and assign priors to the parameters. Pystan also provides us different algorithms to make an inference. It is obvious that we cannot obtain a concise form of the posterior distribution because the posterior form does not belong to any exponential family except the last step. Thus, we resort to the M-H algorithm for the updating of parameters θ_i, α_j, β_j. Finally, we use the updated inverse gamma distribution to update σ_i. Although straightforward, this algorithm is very inefficient and time consuming because of the low accepting rate.

2) Gibbs sampling algorithm. We employ the Gibbs sampling algorithm, a special M-H algorithm with a proposal acceptance rate of 1, in the hope of improving the efficiency. The probability of a correct classification stays the same as

$$P(Y_{ij}=1) = \phi(\alpha_j\theta_i - \beta_j) + \gamma_j(1-\phi(\alpha_j\theta_i-\beta_j)).$$

In order to make inference, we use the data augmentation method[29] in the likelihood function so as to make it easier to analyze. In many cases when the likelihood function cannot be closely approximated by the normal likelihood, the data augmentation method can simplify it by introducing a series of latent variables[30, 31].

Introducing latent variables

In our problem, we define two n by m matrix variables W, Z which are associated with the generating process of the performance matrix. Before explaining them, we make Assumption 3.

Assumption 3. If a sample point falls in the stable region, which is constructed by the classifier, and is not close to the boundary, the classifier can correctly classify the sample with probability 1.

$$W_{ij} = \begin{cases}1, & \text{if sample } j \text{ within classifier } i \text{ is in stable region}\\ 0, & \text{if sample } j \text{ within classifier } i \text{ is in unstable}\\ & \text{boundary.}\end{cases}$$

$Z_{ij} \sim N(\eta_{ij}, 1)$, $\eta_{ij} = \alpha_j\theta_i - \beta_j$. Z serves as a latent indicator variable of W.

For the relation between W and Z, we have the following definition:

$$W_{ij} = \begin{cases}1, & \text{if } Z_{ij} \geq 0\\ 0, & \text{if } Z_{ij} < 0.\end{cases}$$

Prior for normal parameters

For the classifier's ability θ_i, we assign a normal distribution $\theta_i \sim N\left(\mu, \sigma^2\right)$[32]. As α_j and β_j are parameters to describe the discrimination and difficulty of problem j, we can stack α_j and β_j up for simplicity and denote the vector as ϕ_j. In accordance with θ_i, we also assign the new vector of a multivariate normal distribution prior[31]:

$$\Phi = \begin{bmatrix}\phi_1, & \phi_2, & \dots, & \phi_m\end{bmatrix}$$

$$\Sigma_\phi = \begin{bmatrix}\sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta\\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2\end{bmatrix}$$

$$\phi_j = \begin{bmatrix}\alpha_j\\ \beta_j\end{bmatrix} \sim N\left(\begin{bmatrix}\mu_{\alpha_j}\\ \mu_{\beta_j}\end{bmatrix}, \Sigma_\phi\right).$$

We also assume the parameter:

$$M_j = \begin{bmatrix}\mu_{\alpha_j}\\ \mu_{\beta_j}\end{bmatrix} \sim N\left(\begin{bmatrix}\tau_{\alpha_j}\\ \tau_{\beta_j}\end{bmatrix}, I\right).$$

For the hyperparameters τ_{α_j} and τ_{β_j}, we assign a value related to the number of people who correctly answer problem j.

Finally, for the guessing parameter γ, we also assign a beta distribution:

$$\Phi\gamma_j \sim \mathrm{Beta}(s,t).$$

Thus, the joint posterior distribution of $(\Theta, \Phi, M, \Gamma, W, Z|y)$ is

$$\begin{aligned}&P(\Theta, \Phi, M, \Gamma, W, Z|y) =\\ &\quad P(y|W,\Gamma)P(W|Z)P(Z|\Theta,\Phi,M)P(\Theta)P(\Phi|M)P(M)P(\Gamma).\end{aligned}$$

Parameter inference

According to the Gibbs algorithm, we can generate the posterior samples by sweeping through each variable. In each iteration, we sample from the conditional distribution with the remaining values fixed at the current value.

Inference on latent variables

The conditional distribution of W_{ij} can be derived as follows:

$$W_{ij} = \begin{cases}\mathrm{Bernoulli}\left(\dfrac{\Phi(\eta_{ij})}{\gamma_j + (1-\gamma_j)\Phi(\eta_{ij})}\right), & \text{if } y_{ij}=1\\ 0 \quad , & \text{if } y_{ij}=0.\end{cases}$$

Similarly, we can obtain the conditional distribution of Z as

$$Z_{ij} = \begin{cases}N(\eta_{ij},1)I(Z_{ij}\geq 0), & \text{if } W_{ij}=1\\ N(\eta_{ij},1)I(Z_{ij}<0), & \text{if } W_{ij}=0.\end{cases}$$

Inference on normal variables

1) For θ_i

Because of the conjugate prior, we have

$$\theta_i \sim N\left(\frac{\sum_{j=1}^{m}(z_{ij}+\beta_j)\alpha_j+\frac{\mu}{\sigma^2}}{\frac{1}{\sigma^2}+\sum_{j=1}^{m}\alpha_j^2}, \frac{1}{\frac{1}{\sigma^2}+\sum_{j=1}^{m}\alpha_j^2}\right).$$

2) For $\Phi_{\cdot j}$,
We denote $\overline{X}$ as follows:

$$\bar{X}=\begin{bmatrix}\theta_1 & -1\\ \theta_2 & -1\\ \vdots & \vdots\\ \theta_n & -1\end{bmatrix}.$$

Then we have

$$\bar{X}\Phi=\begin{bmatrix}\eta_{11} & \eta_{12} & \cdots & \eta_{1m}\\ \eta_{21} & \eta_{22} & \cdots & \eta_{2m}\\ \vdots & \vdots & \ddots & \vdots\\ \eta_{n1} & \eta_{n2} & \cdots & \eta_{nm}\end{bmatrix}.$$

Thus, we have

$$P(\Phi_{\cdot j}|\sim)\sim N\left(\mu_{\Phi_{\cdot j}},\sigma_{\Phi_{\cdot j}}\right)$$

where $\mu_{\Phi_{\cdot j}}=\left(\bar{X}^{\mathrm{T}}\bar{X}+\Sigma_{\phi}^{-1}\right)^{-1}\left(\bar{X}^{\mathrm{T}}Z_{\cdot j}+\Sigma_{\phi}^{-1}M_j\right), \sigma_{\Phi_{\cdot j}} = \left(\bar{X}^{\mathrm{T}}\bar{X}+\Sigma_{\phi}^{-1}\right)^{-1}$.

3) For M_j,

$$P(M_j|\sim)\sim N\left(\mu_{M_j},\sigma_{M_j}\right)$$

where $\mu_{M_j}=\left(\Sigma_{\phi}^{-1}+I\right)^{-1}\left(\Sigma_{\phi}^{-1}\Phi_{\cdot j}+T_j\right), \sigma_{M_j}= \left(\Sigma_{\phi}^{-1}+I\right)^{-1}$.

4) For γ_j,

$$\gamma_j\sim \text{Gamma}\left(\lambda_1,\lambda_2\right)$$

where $\lambda_1=\sum\limits_{j=1}^{m}\sum\limits_{i=1}^{n}I(W_{ij}=0,y_{ij}=1)+s, \lambda_2=\sum\limits_{j=1}^{m}\sum\limits_{i=1}^{n}I(W_{ij}=0,y_{ij}=0)+t$.

4 Bernoulli-beta model

4.1 Bernoulli-beta model construction

In the previous model, we quantify the classifier's ability by defining the probability of a correct classification, and make an assumption about the parameters for simplicity of calculation in the MCMC approach. However, this method is limited in that it confines the distribution of a correct classification, which cannot be directly measured by the performance matrix. To obtain a closed form, we introduce many normal assumptions for the relationship between latent variables. Besides, Gibbs sampling is also time consuming. Thus, instead of assuming a certain distribution for the latent variables, a new model is proposed in which a constraint relaxation is applied to the latent parameters. In this model, the successful prediction of sample j is also determined by its difficulty and the ability of classifier i. Now, we consider the generating process of each element Y_{ij} in performance matrix Y. For each element Y_{ij}:

$$\begin{aligned}Y_{ij}&\sim \text{Bernoulli}(P_{ij})\\ P_{ij}&\sim \text{Beta}(m_{ij},n_{ij}).\end{aligned}$$

Based on the Bernoulli-Beta conjugate distribution, we can see the value of Y_{ij} is strongly associated with a probability P_{ij}, and we define it as the successful parameter. It is clear that the larger the P_{ij} is, the higher the probability of $Y_{ij}=1$ is, meaning that the classifier i correctly classifies sample j. Their relationship can be shown as

$$\begin{aligned}&P(Y_{ij}=1)=P_{ij}\\ &f(P_{ij})=\frac{\Gamma(m_{ij}+n_{ij})}{\Gamma(m_{ij})\Gamma(n_{ij})}P_{ij}^{m_{ij}-1}(1-P_{ij})^{n_{ij}-1}.\end{aligned}$$

Thus, we want the successful parameters to be increasing functions of $\Delta_{ij}=\theta_i-\beta_j$ and keep the parameters m_{ij} and n_{ij} positive. We need to construct a special relationship to link them with θ_i and β_j. We can construct a function:

$$\begin{aligned}m_{ij}&=\exp\left\{\frac{\alpha_j\theta_i-\beta_j}{2}\right\}\\ n_{ij}&=\exp\left\{\frac{-\alpha_j\theta_i+\beta_j}{2}\right\}.\end{aligned}$$

The exponential function ensures that m_{ij} and n_{ij} can take on positive values. With this structure, the success probability can be indirectly affected by the classifier's ability and the sample's difficulty. Most importantly, no assumptions are made for the classifier's ability[33].

We can also view the relationship in a different perspective. If we find the expectation of P_{ij} under the assumption and expand it as a function of parameters θ_i and β_j, the following sigmoid function will appear.

$$E(P_{ij})=\frac{m_{ij}}{n_{ij}+m_{ij}}=\frac{1}{1+\exp\left\{-\alpha_j\theta_i+\beta_j\right\}}.$$

Obviously, we bridge the latent parameter P_{ij}, θ_i, β_j in a more flexible approach than the previous model. Their relation is no longer defined by a constant equation, but a distribution with a sigmoid function. Although the relation here has no distributional interpretation, it quantifies how the distance between θ_i and β_j contributes to the successful probability, and then generates a prediction result[34]. Moreover, θ_i and β_j are all free variables without a strong distributional constraint. Fig. 1 displays the expected response function with different difficulties and discriminations.

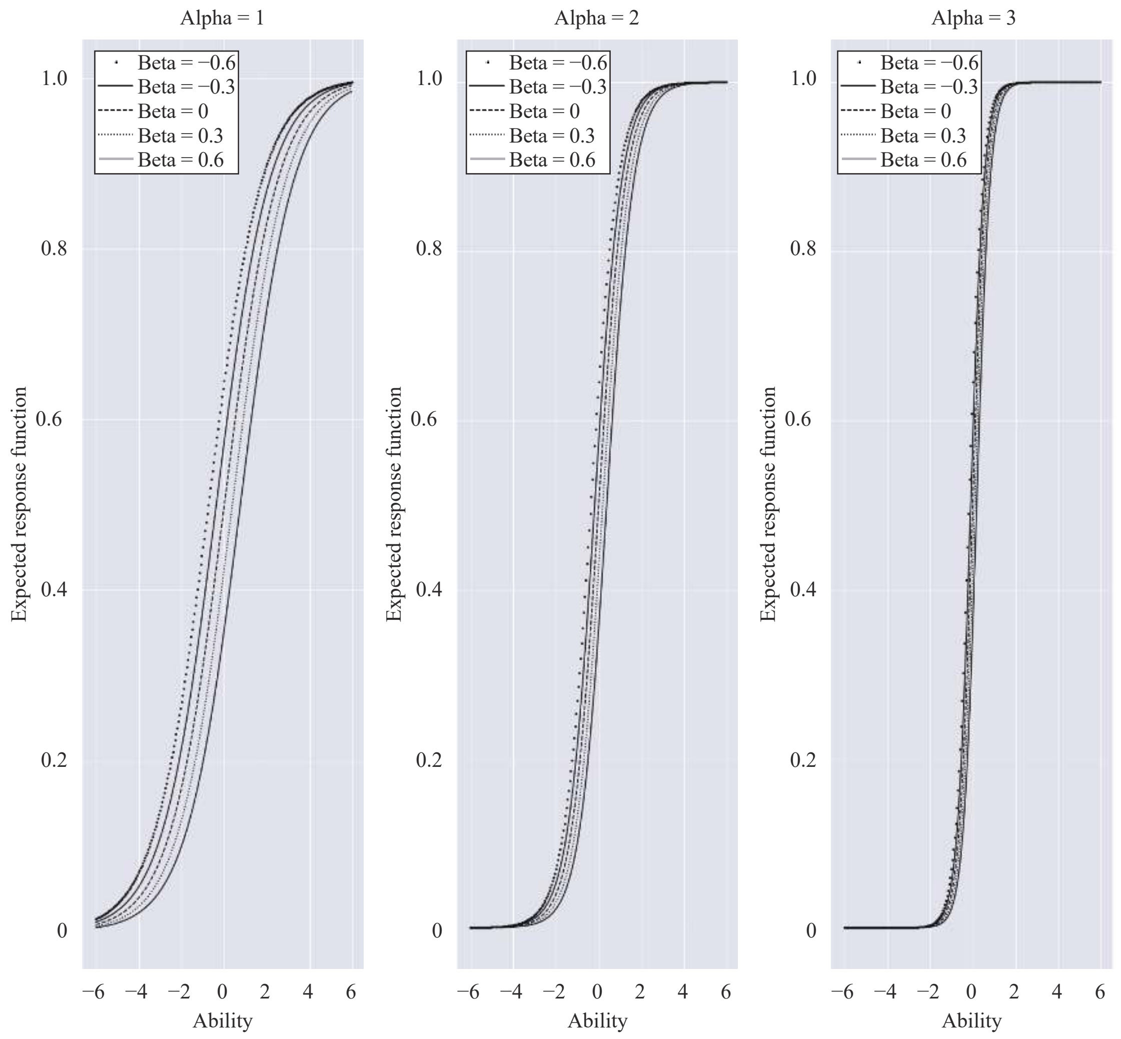


Fig. 1 The expected response function with different difficulty and discrimination. The higher the discrimination is, the steeper the curve is. The difficulty can affect the probability for correctly classifying the sample.

Now we can summarize the full model log joint probability as follows:

$$L(Y,P,M,N)=\sum_{i=1}^{n}\sum_{j=1}^{m}(y_{ij}+m_{ij}-1)\ln(P_{ij})+ (n_{ij}-y_{ij})\ln(1-P_{ij})+\ln\left(\frac{\Gamma(m_{ij}+n_{ij})}{\Gamma(m_{ij})\Gamma(n_{ij})}\right)=\sum_{i=1}^{n}\sum_{j=1}^{m}L_{ij}.$$

We denote all the parameters m_{ij} and n_{ij} as

$$\Theta=\bigcup_{i,j}(m_{ij},n_{ij}).$$

Now the model is constructed.

4.2 Parameter inference

A more difficult, but common, situation is that we introduce a latent parameter P_{ij} while making no distributional assumption about parameters θ_i and β_j. To solve this problem, we adopt the expectation maximization (EM) algorithm[35] which is a standard tool for the maximum likelihood algorithm with latent variables[36]. We notice that the parameters are redundant because the likelihood function only depends on the distance of θ_i and β_j. Thus, a constraint for the parameters is necessary. We set the mean of β_j equals to 0, i.e., $\sum_{j=1}^{m}\beta_j=0$.

E-step

The E-step, on the $(k+1)$-th iteration, requires the calculation of

$$Q(\Theta|\Theta^k)=E_{\Theta^k}[L(\Theta,P|Y)]$$

$$Q(L|M,N,Y)=\sum_{i=1}^{n}\sum_{j=1}^{m}Q(L_{ij}|m_{ij}^k,n_{ij}^k,y_{ij})=\sum_{i=1}^{n}\sum_{j=1}^{m}E_{m_{ij}^k,n_{ij}^k}[L_{ij}|y_{ij}]=\sum_{i=1}^{n}\sum_{j=1}^{m}Q_{ij}.$$

From the full probability function above, we can derive the posterior distribution of P_{ij}:

$$P(P_{ij}|m_{ij},n_{ij},y_{ij}) \propto P_{ij}^{y_{ij}+m_{ij}^k-1}(1-P_{ij})^{n_{ij}^k-y_{ij}} \sim \text{Beta}\left(y_{ij}+m_{ij}^k, n_{ij}^k-y_{ij}+1\right).$$

Therefore, Q can be expressed as

$$\begin{aligned}
&Q(L|M,N,Y) = \\
&\sum_{i=1}^{n}\sum_{j=1}^{m} Q_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{m} E_{m_{ij}^k,n_{ij}^k}[L_{ij}|y_{ij}] = \\
&\sum_{i=1}^{n}\sum_{j=1}^{m} (y_{ij}+m_{ij}-1)\psi\left(y_{ij}+m_{ij}^k\right) + \\
&\sum_{i=1}^{n}\sum_{j=1}^{m} (n_{ij}-y_{ij})\psi\left(n_{ij}^k-y_{ij}+1\right) - \\
&\sum_{i=1}^{n}\sum_{j=1}^{m} (m_{ij}+n_{ij}-1)\psi\left(n_{ij}^k+m_{ij}^k+1\right)
\end{aligned}$$

where $\psi(x)=\frac{\Gamma(x)'}{\Gamma(x)}$ is a digamma function.

M-step

As we have the constraint $\sum_{j=1}^{m}\beta_j=0$, we can express the Q function as follows:

$$\begin{aligned}
&Q(L|M,N,Y) = \\
&\sum_{i=1}^{n}\sum_{j=1}^{m-1} m_{ij}SM_{ij} + \sum_{i=1}^{n} SM_{im}\left(\frac{\alpha_j\theta_i+\sum_{j=1}^{m-1}\beta_j}{2}\right) + \\
&\sum_{i=1}^{n}\sum_{j=1}^{m-1} n_{ij}SN_{ij} + \sum_{i=1}^{n} SN_{im}\left(\frac{-\alpha_j\theta_i-\sum_{j=1}^{m-1}\beta_j}{2}\right)
\end{aligned}$$

where $SM_{ij}=\psi(y_{ij}+m_{ij}^k)-\psi(n_{ij}^k+m_{ij}^k+1)$, $SN_{ij}=\psi(n_{ij}^k-y_{ij}+1)-\psi(n_{ij}^k+m_{ij}^k+1)$.

To estimate the parameters, we iterate over the set of parameters until they converge. We can use different optimization methods such as gradient ascent, RMSProp or Adam. Here we show the algorithm for gradient ascent. The partial derivative of Q with respect to θ_i and β_j is as follow:

$$\begin{aligned}
\frac{\partial Q}{\partial\theta_i} &= \sum_{j=1}^{m}\frac{1}{2}m_{ij}SM_{ij}-\frac{1}{2}n_{ij}SN_{ij} \\
\frac{\partial^2 Q}{\partial\theta_i^2} &= \sum_{j=1}^{m}\frac{1}{4}m_{ij}SM_{ij}+\frac{1}{4}n_{ij}SN_{ij} \\
\frac{\partial Q}{\partial\beta_j} &= \sum_{i=1}^{N}\left(-\frac{1}{2}m_{ij}SM_{ij}\right)+\frac{1}{2}m_{im}SM_{im}+ \\
&\quad \frac{1}{2}n_{ij}SN_{ij}-\frac{1}{2}n_{im}SN_{im} \\
\frac{\partial^2 Q}{\partial\beta_j^2} &= \sum_{i=1}^{N}\frac{1}{4}m_{ij}SM_{ij}+\frac{1}{4}m_{im}SM_{im}+ \\
&\quad \frac{1}{4}n_{ij}SN_{ij}+\frac{1}{4}n_{im}SN_{im}.
\end{aligned}$$

Based on the partial derivatives, we can use the gradient ascent algorithm to update the parameters Θ in each iteration.

$$\begin{aligned}
\theta_i^{k+1} &= \theta_i^k + \alpha\frac{\partial Q}{\partial\theta_i}_{\Theta^k} \\
\beta_j^{k+1} &= \beta_j^k + \alpha\frac{\partial Q}{\partial\beta_j}_{\Theta^k}.
\end{aligned}$$

5 Assigning weights to classifiers

As the estimators of θ reflect classifiers' ability, we should assign classifier i a weight according to the estimation of θ_i. The estimator, however, can be either positive or negative. Thus, we employ the following transformation to decide the value of weight:

$$w_i = \frac{\mathrm{e}^{\theta_i}}{\sum_{k=1}^{n}\mathrm{e}^{\theta_k}}.$$

The weights clearly illustrate the order of classifiers' ability and are normalized by the above formula.

6 Results

6.1 Results of the inference

A simulation study is presented below to show the estimation of the parameters from two 10 by 1 000 datasets, which are generated from different parameter settings. The difference lies in the parameters' distributions. For simplicity, we denote the normal IRT model estimated by the M-H algorithm, by Gibbs sampling algorithm and the Bernoulli-Beta IRT model as Models 1, 2 and 3, respectively, and our experiments focus more on Models 2 and 3. First, we apply the three models to estimate the parameters of the two datasets generated from two different sets of parameters. Both settings contain 10 samples with 1 000 classifiers, which have fixed difficulty and ability. In the first dataset, all parameters are sampled from a normal distribution so that they meet the assumption of the first two models. However, the distributions of the parameters in the second dataset vary. The difficulty is manually set to have a large variance and a relatively large range, and the ability is sampled from a highly skewed gamma distribution that the second dataset cannot satisfy the first two models' assumption. Thus, we can compare the accuracies of the two models in different situations.

Four measures were computed to evaluate the performance of the three models. θ is the real parameter and θ^{est} is an estimate of the parameter:

1) Correlation:

$$\text{Corr} = \frac{N\sum_{i=1}^{N}\theta_i^{\text{est}}\theta_i - \sum_{i=1}^{N}\theta_i^{\text{est}}\sum_{i=1}^{N}\theta_i}{N\sigma_{\text{est}}\sigma}$$

2) Mean square error:

$$\text{MSE} = \frac{\sum_{i=1}^{N}(\theta_i^{\text{est}} - \theta_i)^2}{N}$$

3) Mean absolute error:

$$\text{MAE} = \frac{\sum_{i=1}^{N}\|(\theta_i^{\text{est}} - \theta_i)\|}{N}$$

4) Variance ratio:

$$\text{VR} = \frac{\sigma_{\text{est}}^2}{\sigma^2}.$$

The correlation is to test the linear relationship between the parameter estimates and the real parameters. The MSE and MAE measure the precision of the estimation and the variance ratio illustrates the comparative stability. In Tables 1 and 2, we list the previous four measures of the estimators (classifiers' abilities and problems' difficulties) obtained from different models. In Tables 3 and 4, we show the real parameters and their estimated values from three different models.

As we can see from the result, models 1 and 2 perform well on the first dataset while model 3 beats the others on the second dataset. The reason is the first two models tend to give a relatively stable solution due to the normal prior, which has the advantage of minimizing the range between the estimators. Thus, when the parameters are generated from a normal distribution, they fit well. In model 3, we didn't have such an assumption, so the parameters can take any value to minimize the loss function and the range may be larger than that of the first two estimations. It clearly produces a better estimation when the ranges of the real parameters are large. To compare the accuracy of the estimation, we calculate the error ratio for each parameter in both datasets:

Table 1 Measures for dataset 1

Model	Parameter	Correlation	MSE	MAE	VR
Model 1	θ (Classifiers' ability)	0.86	0.97	0.96	0.97
Model 2	θ (Classifiers' ability)	0.88	0.31	0.37	0.88
Model 3	θ (Classifiers' ability)	0.87	0.91	0.96	1.9
Model 1	β (Samples' difficulty)	0.97	0.007	0.16	1.15
Model 2	β (Samples' difficulty)	0.99	0.003	0.05	0.99
Model 3	β (Samples' difficulty)	0.99	0.016	0.12	1.36

Table 2 Real and estimated parameters for dataset 1

Parameter component	Parameters value	Model 1 estimation	Model 2 estimation	Model 3 estimation
β_1	−1.024	−0.94	−0.953	−1.107
β_2	−0.934	−0.99	−0.91	−1.08
β_3	−0.694	−0.732	−0.709	−0.869
β_4	−0.464	−0.56	−0.514	−0.579
β_5	0.356	0.472	0.408	0.503
β_6	0.336	0.336	0.319	0.382
β_7	0.616	0.694	0.576	0.722
β_8	0.806	0.85	0.757	0.956
β_9	−0.074	−0.171	−0.145	−0.172
β_{10}	1.076	1.23	1.141	1.243

Table 3 Measures for dataset 2

Model	Parameter	Correlation	MSE	MAE	VR
Model 1	θ (Classifiers' ability)	0.63	23	4.2	0.87
Model 2	θ (Classifiers' ability)	0.87	21.22	3.9	0.81
Model 3	θ (Classifiers' ability)	0.93	19.33	3.1	0.95
Model 1	β (Samples' difficulty)	0.86	15.75	3.53	0.8
Model 2	β (Samples' difficulty)	0.93	12.64	2.58	0.18
Model 3	β (Samples' difficulty)	0.98	1.716	1.13	1.08

Table 4 Real and estimated parameters for dataset 2

Parameter component	Parameters value	Model 1 estimation	Model 2 estimation	Model3 estimation
β_1	−9.075	−4.35	−3.256	−9.824
β_2	−2.485	−1.3	−1.296	−2.68
β_3	−3.395	−1.2	−2.24	−3.85
β_4	−1.015	−2.2	−0.192	−0.203
β_5	−0.075	0.461	1.593	1.248
β_6	0.165	1.36	0.96	1.628
β_7	4.035	3.194	3.045	5.591
β_8	6.485	3.85	3.783	6.998
β_9	−6.395	−3.171	−2.904	−8.462
β_{10}	11.755	3.23	3.534	9.553

$$\text{Error ratio} = \frac{(\theta^{\text{est}} - \theta)^2}{\text{Average MSE}}$$

$$\text{Average MSE} = \frac{\text{MSE}_1 + \text{MSE}_2 + \text{MSE}_3}{3}.$$

In Fig. 2, we use a bar plot to compare the error ratios for all models on two datasets.

From the results, we can make a conclusion that real parameters and parameter estimates are highly correlated. For both datasets, the correlations are almost always higher than 0.85, implying that the parameter estimates hold a strong linear relationship with the true parameter values. Thus, it makes more sense to keep the weights of different classifiers as ordinal consistent with the classifiers' ability[37].

6.2 Analysis of difficulty parameters

As we mentioned before, an IRT ensemble can evalu-

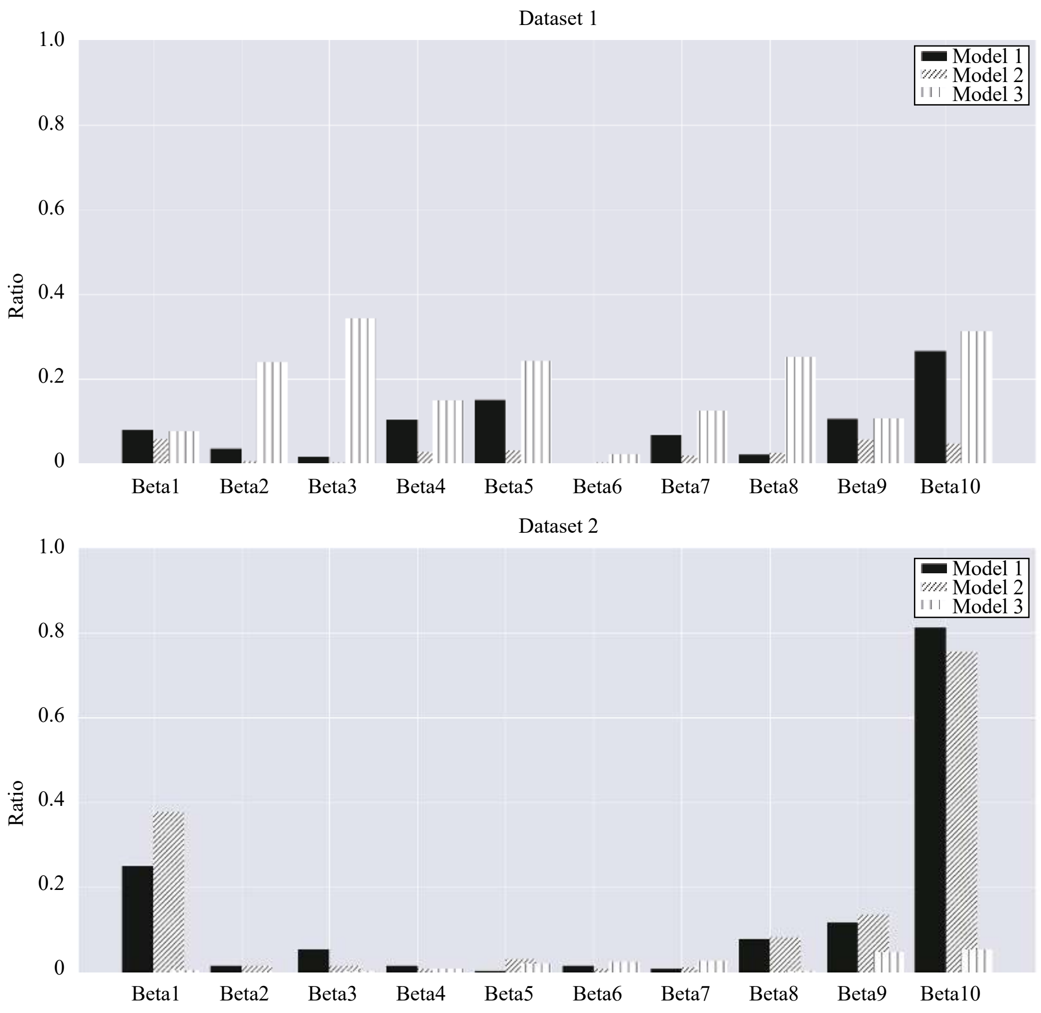


Fig. 2 Error ratio for the two datasets

ate the classifiers' ability and samples' difficulty simultaneously. To better understand the model, we first show what the parameters can reflect, and how they can affect the classification. Then we will illustrate the performance of our method on different datasets.

We depict the sample points from the chess board dataset to show how the IRT ensemble model evaluates samples' difficulty. Figs. 3 and 4 illustrate the dual character of location and difficulty, which is shown by the size of the sample points. The larger the point is, the larger the difficulty of parameter is. Points in the same block share the same color in the original chess board. We first show how the estimated values change with increased iterations. In Fig. 3, we constructed 500 base classifiers. As we increase the number of iterations, difference is gradually increased. The outcome of the 100 iterations is similar, while the outcome of the 500 iterations is various, meaning each sample is well distinguished by its difficulty evaluated by a bunch of classifiers. Zooming in to Fig. 3, we can find some rules. The IRT ensemble method tends to assign a higher difficulty to the points that are closer to the boundary, and these points support the decision boundary in return. It is obvious that when the points are close to their counterparts with a distinct label, they are more likely to be misclassified. Only those classifiers which are powerful enough can correctly complete the task of difficult classification. Thus, it makes sense to take their capability for constructing the classification boundary.

For some other ensemble methods, it proves that increasing the number of classifiers can improve the performance, which also applies to the IRT ensemble method. In Fig. 4, we fixed the number of iterations to be 2 000 and changed the number of classifiers. According to the experiment, when we increase the number of classifiers, those sample points constructing the boundary within the block will stand out while those inside the boundary will shrink to a dot. In order to distinguish the important sample points from others, more classifiers should be included to make a joint decision. The interesting bit comes when we increase the number of classifiers to a large enough value. In the last two subplots, the sizes of the points seem to be unchanged. In many cases, increasing the decision size cannot guarantee improved performance.

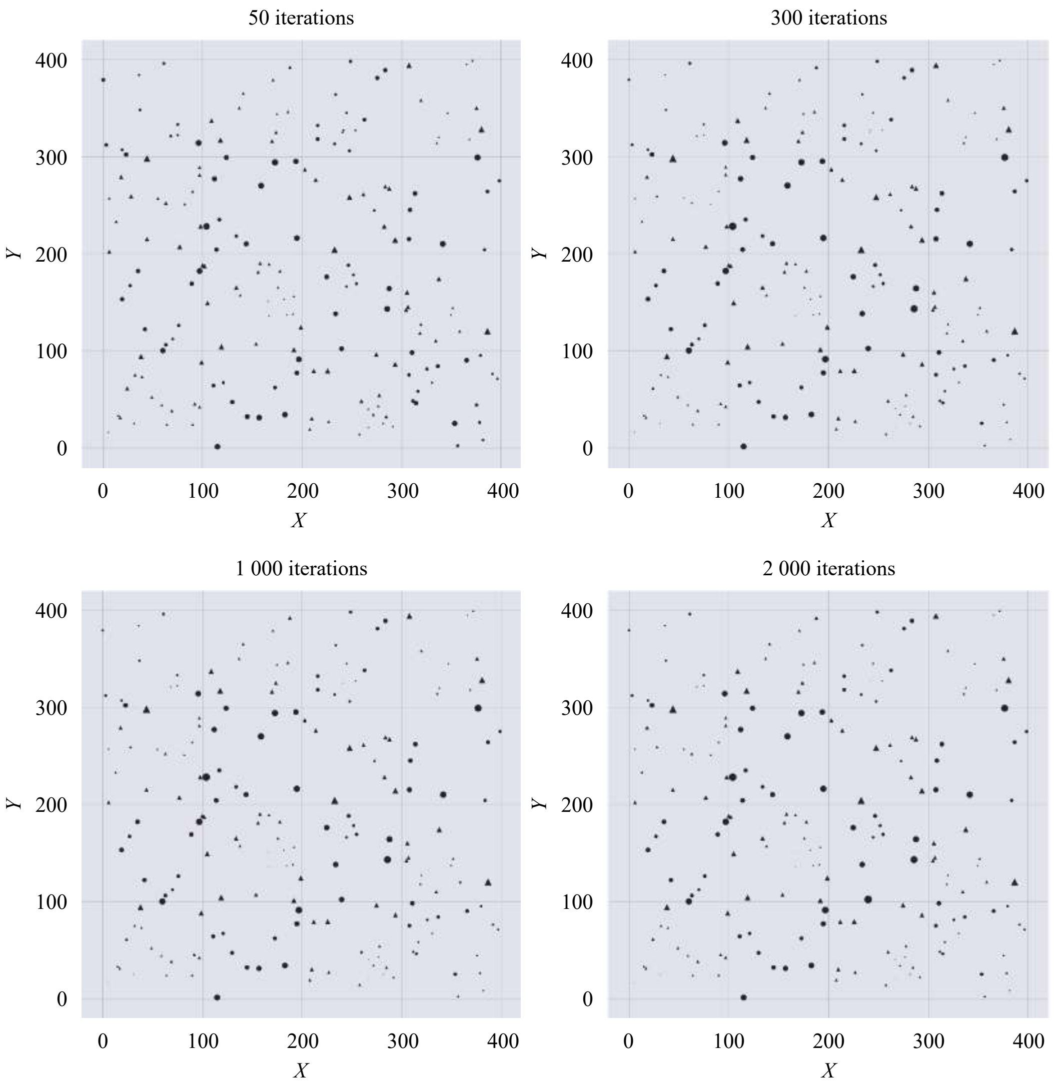


Fig. 3 500 Classifiers with different iterations. X and Y axes illustrate the position of the points, and different class labels are shown by distinct shape.

When we have sufficient classifiers, it will come across the bottleneck.

6.3 Analysis of classification

We collected 15 real datasets and 2 artificial datasets to compare our Model 2 with a single tree[38], random forest (RF)[39], bagging[7], gradient boosting decision tree (GBDT)[40], linear discriminant analysis (LDA)[41], and support vector machine (SVM)[42]. We didn't make experiments for Model 1 because it is time consuming. For all the ensemble methods, we used a single classification tree as the base classifier. When it comes to a single tree, the pruning option is necessary for preventing overfitting. However, we didn't implement the pruning algorithm for the base classifiers in bagging, gradient boosting and random forest because pruning decreases the variance but increases the bias. For all the ensemble algorithms, bootstrap can be used to construct various base tree structures, which can reduce the variance effectively. Thus in all the ensemble models, we used unpruned trees as the base classifiers to account for bias and then used bootstrap to reduce the variance.

Most of the datasets summarized in Table 5 are from the UCI datasets. As our model is constructed using the base classifiers, it is suitable for all kinds of features as long as the base model is adequate for the data. In order to show a generalization of our method, we intentionally selected some datasets containing both the categorical features as well as the continuous features. One-hot encoding is also a must for all the nominal features. We conducted all the experiments on Python 3.6 platform. To compare the accuracy of these methods, we randomly split the dataset to 10 folds and set the test set proportional to 0.3. A simulation of each setting was performed 30 times for each dataset. In order to compare the ac-

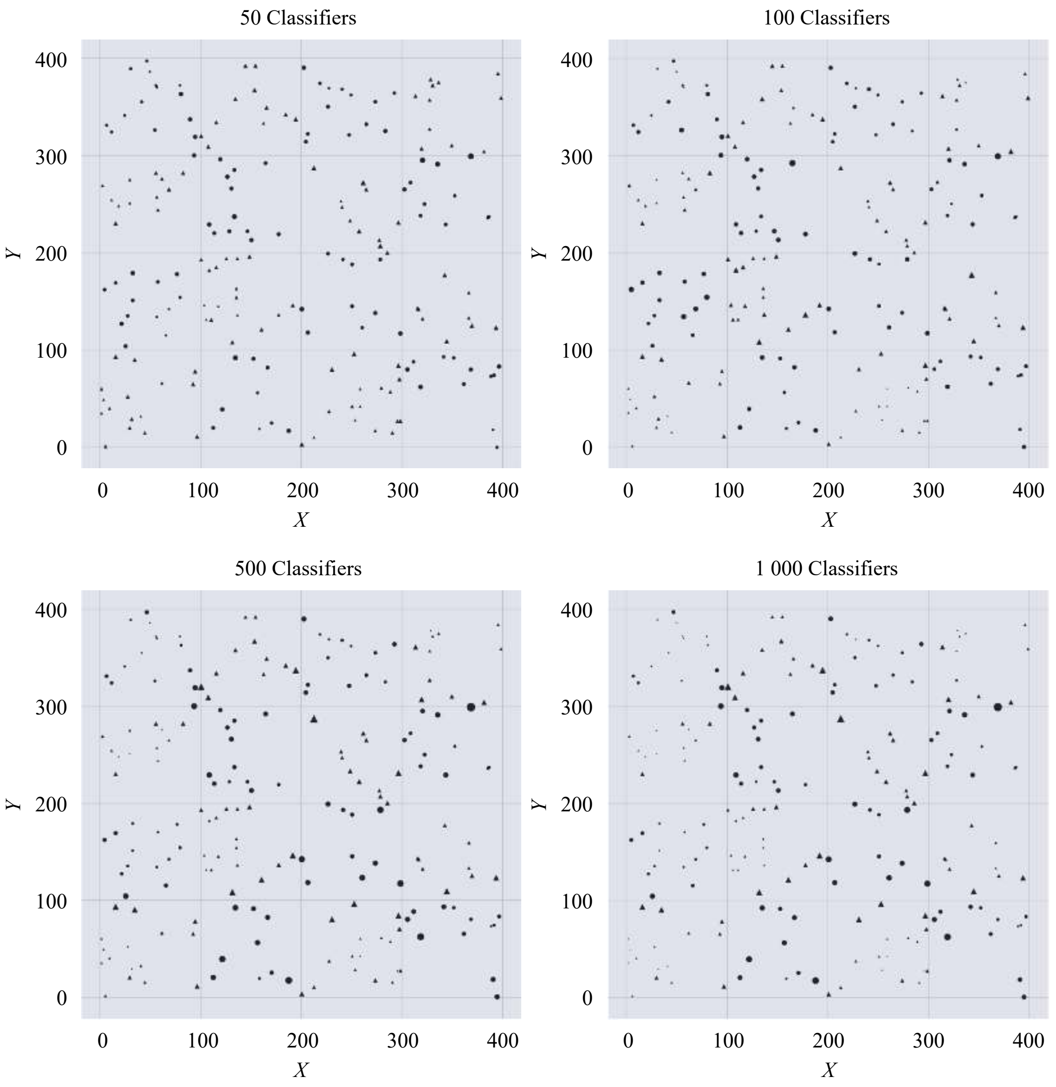


Fig. 4 Different number of classifiers. X and Y axes illustrate the position of the points, and different class labels are shown by distinct shape.

curacies of various methods, we set the number of trees in each ensemble algorithm to 500.

The result of the average accuracy is in Table 6. We highlighted the best two results and the worst result for each data set. From Table 6, it seems that for Model 2, random forest and gradient boosting perform well in general. However, for some kinds of data sets, gradient boosting fails to recognize that pattern and yields the worst result[43]. The weakness of gradient boosting has been reported in some papers. The performance of SVM greatly fluctuated[44].

Bagging generally performs better than the tree model. Although bagging is not the best, it is more stable than gradient boosting in some cases. It is noted that Model 2 is within the scope of a weighted voting model, which extends from the bagging strategy. Thus, we can explain the reason why Model 2 is more stable than the gradient boosting method.

A win table[45] summarizes the comparison in Table 7. In the win table, $a_{i,j}$ illustrates the frequency that method j gives a higher accuracy than method i. For instance, $a_{1,3}$ equals 11 means in total 17 comparisons between Model 2 and gradient boosting, Model 2 produces more accurate or the same result than gradient boosting in 11 datasets. This table shows every pairwise comparison in detail. In order to rank the methods, we need to calculate the goal difference from the win table, subtracting the frequency of loss from the frequency of wins for each model. The frequency of wins can be obtained by summing within the row, while the frequency of losses can be obtained within the column for each model. From there, the goal difference can be calculated. The result is shown in Table 8. It is clear that Model 2 has an overwhelming superiority over others. This table provides another way to show the consistently high accuracy of our model compared to other methods shown in Table 6.

We also conducted an experiment to investigate how the ensemble size affects the prediction with 13 datasets

Table 5 Dataset information

Dataset	Observations	Continuous features	Discrete features	Class	Source
IRIS	150	4	0	3	UCI
Bld	345	6	0	2	UCI
Spe	267	44	0	2	UCI
Glass	214	9	0	6	UCI
Veh	846	18	0	4	UCI
Checkboard	160 000	2	0	2	Artificial
BTD	106	9	0	4	UCI
IPLD	583	8	1	2	UCI
Haberman	306	3	0	2	UCI
Ionos	351	32	2	2	UCI
Multiangle	160 000	2	0	2	Tensorflow
Balance	625	0	4	2	UCI
AUS	690	5	9	2	UCI
ECOLI	336	7	0	6	UCI
LEN	24	0	3	3	UCI
TAE	151	0	5	3	UCI
LC	33	0	56	3	UCI
LSVT	126	310	0	2	UCI
SCADI	70	205	0	8	UCI

Table 6 Average of accuracy

DataSet	Model 2	RF	GBDT	SVM	Single tree	Bagging	LDA
IRIS	**0.967**	**0.962**	0.945	0.932	0.947	0.954	**0.88**
Bld	**0.703**	0.69	**0.693**	0.68	0.62	0.66	**0.59**
Spe	**0.91**	**0.91**	0.88	**0.828**	0.861	**0.911**	0.865
Glass	0.729	**0.766**	**0.77**	0.63	0.72	0.72	**0.59**
Veh	0.74	0.73	**0.75**	**0.76**	0.63	0.68	**0.53**
Checkboard	**0.88**	0.82	**0.96**	**0.5**	0.58	0.87	**0.5**
BTD	0.85	0.875	**0.88**	**0.89**	**0.82**	0.86	0.85
IPLD	**0.723**	**0.712**	0.7	0.71	**0.68**	0.7	**0.68**
Harman	0.677	0.668	**0.65**	**0.74**	0.71	**0.73**	**0.73**
Ionos	**0.928**	**0.926**	0.92	0.886	0.888	0.923	**0.855**
Multiangle	**0.92**	0.9	**0.96**	0.51	0.83	0.83	**0.49**
Balance	**0.815**	**0.849**	0.623	0.78	**0.59**	0.65	0.73
Aus	**0.858**	**0.865**	0.81	0.76	0.82	0.83	**0.68**
ECOLI	**0.833**	0.755	**0.65**	**0.87**	0.8	0.81	0.829
Len	**0.738**	**0.706**	**0.725**	0.717	0.708	0.72	**0.725**
TAE	**0.565**	0.468	**0.582**	0.511	**0.37**	0.43	0.44
LC	**0.5**	**0.484**	**0.5**	0.45	0.43	0.43	**0.2**

(4 datasets were discarded because of the failure of computation when the ensemble size is too small). We still used the same method for getting the accuracy and 20 repetitions were conducted for each sample size, which are averaged to calculate the t statistics. In Fig. 5, we show

Table 7 Win table

Method	Model 2	RF	GBDT	SVM	Single tree	Bagging	LDA
Model 2	0	13	11	13	16	14	16
RF	5	0	8	11	14	12	14
GBDT	7	9	0	11	13	10	14
SVM	4	6	6	0	10	9	15
Tree	1	3	4	7	0	3	10
Bagging	3	5	8	8	17	0	13
LDA	2	3	4	3	8	5	0

Table 8 Summary of win table

Models	WinLoss	Win	Loss
Model 2	61	83	22
RF	25	64	39
GBDT	23	64	41
SVM	−3	50	53
Tree	−50	28	78
Bagging	1	54	53
LDA	−57	25	82

the boxplot. When comparing our method with random forest, gradient boosting and SVM, we conclude that the gain of our model tends to be enhanced as the ensemble size increases. It seems that our model has a good potential to improve the accuracy if the ensemble size is large enough.

Model 3 has predominant advantage when applying to some datasets. We illustrate the cumulative accuracy on 4 cases in Fig. 6. In these datasets, samples contain many attributes compared to the sample size, which means they include a lot of redundant information. Consequently, the samples' difficulties vary and there may exist a small subsample contributing a lot for constructing the decision boundary. It is hard for most of the classifiers to detect the important variables. Only a small subset of classifiers are powerful. Thus, the distribution of the classifiers are no longer symmetric and the variance of the classifiers' abilities increases. Model 3 is more powerful than Model 2 when the variance of the parameters is large enough, higher weights are assigned for the stronger classifiers. Thus, it can outperform other methods in these cases. We found that Model 3 can consistently produce a higher accuracy than random forest and gradient boosting. However, when the difficulty of each sample is similar, Model 2 tends to perform better.

7 Conclusions

In this paper, we proposed the IRT ensemble, a weighted majority voting method focusing on the classifiers that can correctly deal with the hard-to-classify problems, by adopting the item response theory. The classify-

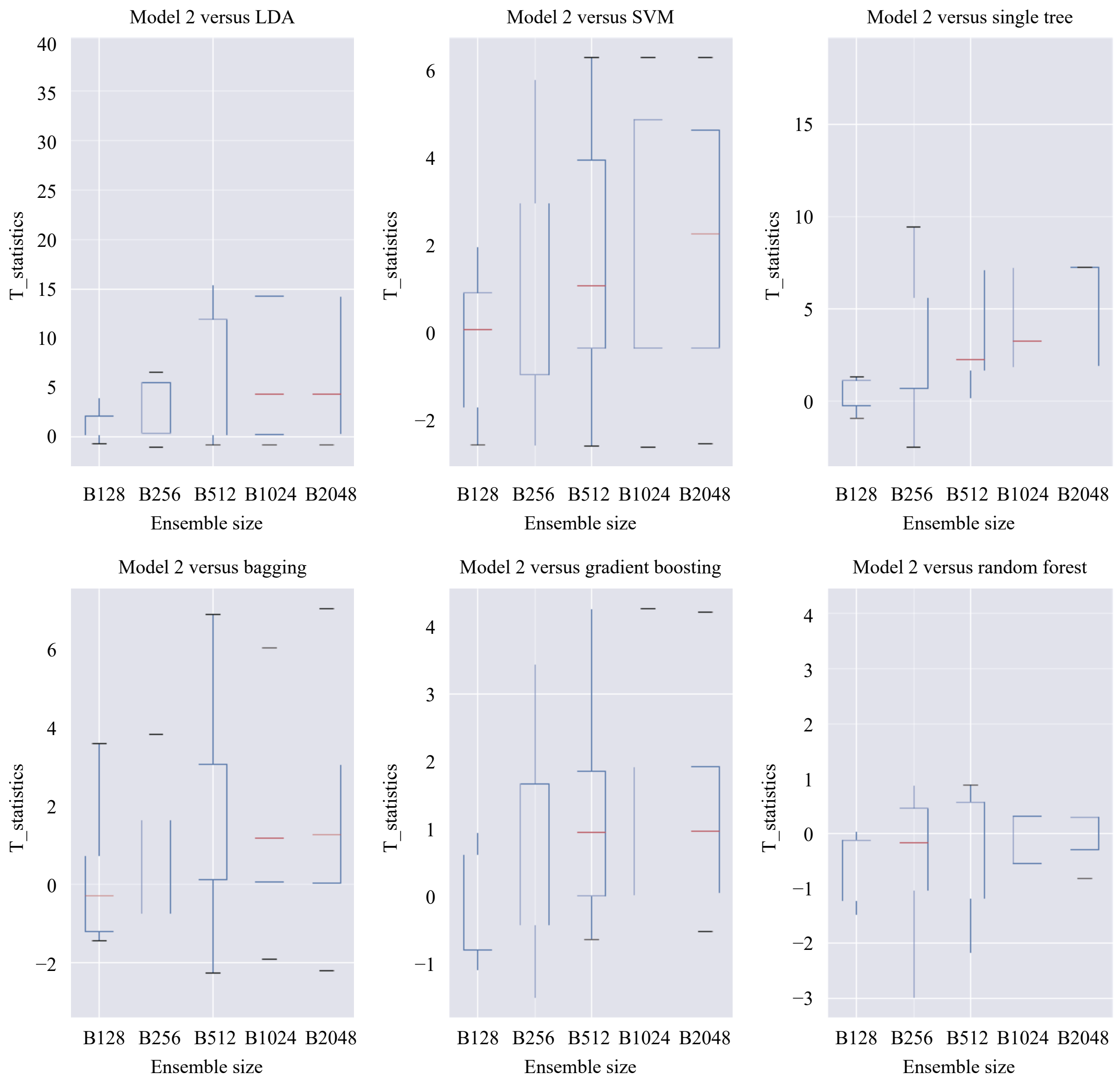


Fig. 5 Comparison with LDA, SVM, single tree, bagging, gradient and random forest

ing boundaries are constructed by the points that are frequently misclassified and higher weights are assigned to the classifiers with higher abilities. We also proposed three models to estimate the ability parameters and introduced the assumptions behind the models.

For the performance of the models, we analyzed them in two stages. First, we evaluated their accuracy in the estimation of parameters. We concluded that Models 1 and 2 perform well when the variance of the parameters are small, while Model 3 is more suitable when the parameters vary. We also explained how the lengths of the Markov chains and the number of classifiers would affect the estimation of samples' difficulty. The chessboard dataset also provides us an intuitive explanation about the idea behind the IRT ensemble algorithm. Finally, we implemented an experiment with Model 2 using 19 datasets and compared the performance with other classification methods. We showed that the advantage of Model 2 is enhanced with the increased ensemble size compared to LDA, SVM, single tree, bagging, and gradient boosting. It showed compatible performance with random forest. Finally, we found Model 3 has an edge in high dimensional datasets.

Future work includes combining Model 3 with kernel methods. Another modification is to introduce the Beta model, which is widely used in the network analysis.

References

[1] Z. H. Zhou. Ensemble learning. *Encyclopedia of Biometrics*, S. Z. Li, Ed., Berlin, Germany: Springer, pp. 411–416, 2009.

[2] L. Lam, S. Y. Suen. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernet-*

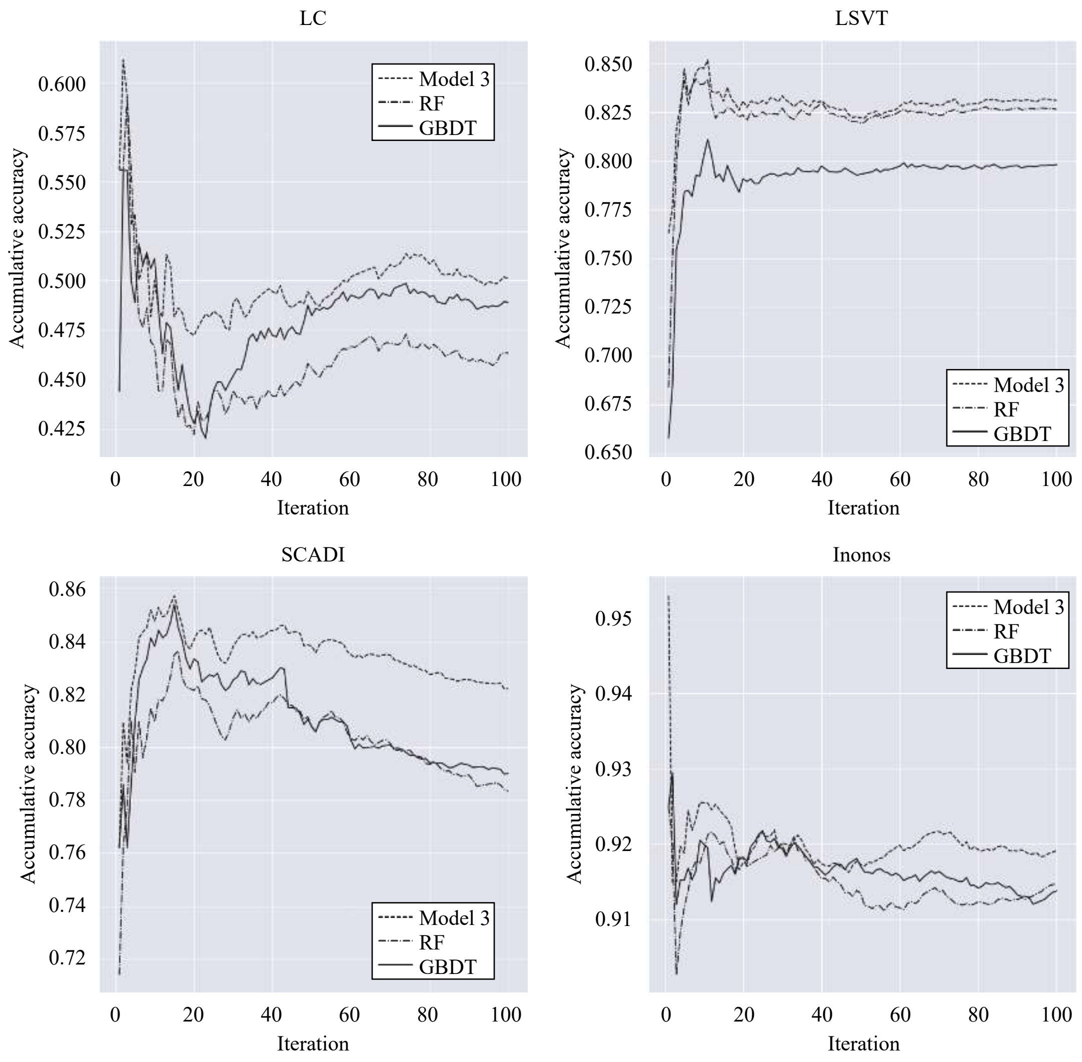


Fig. 6 Cumulative accuracy for Model 3 on 4 different datasets

ics – Part A: *Systems and Humans*, vol. 27, no. 5, pp. 553–568, 1997. DOI: 10.1109/3468.618255.

[3] A. F. R. Rahman, H. Alam, M. C. Fairhurst. Multiple classifier combination for character recognition: revisiting the majority voting system and its variations. In *Proceedings of the 5th International Workshop on Document Analysis Systems*, pp. 167–178, Springer, Princeton, USA, 2002.

[4] H. Kim, H. Kim, H. Moon, H. Ahn. A weight-adjusted voting algorithm for ensembles of classifiers. *Journal of the Korean Statistical Society*, vol. 40, no. 4, pp. 437–449, 2011. DOI: 10.1016/j.jkss.2011.03.002.

[5] S. E. Embretson, S. P. Reise. *Item Response Theory*, New York, USA: Psychology Press, 2013.

[6] F. Martínez-Plumed, R. B. C. Prudêncio, A. Martínez-Usó, J. Hernández-Orallo. Item response theory in AI: Analysing machine learning classifiers at the instance level. *Artificial Intelligence*, vol. 271, pp. 18–42, 2019. DOI: 10.1016/j.artint.2018.09.004.

[7] L. Breiman. Bagging predictors. *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996. DOI: 10.1007/BF00058655.

[8] I. Gandhi, M. Pandey. Hybrid ensemble of classifiers using voting. In *Proceedings of International Conference on Green Computing and Internet of Things*, IEEE, Noida, India, pp. 399–404, 2015. DOI: 10.1109/ICGCIoT.2015.7380496.

[9] A. Rojarath, W. Songpan, C. Pong-Inwong. Improved ensemble learning for classification techniques based on majority voting. In *Proceedings of the 7th IEEE International Conference on Software Engineering and Service Science*, IEEE, Beijing, China, pp. 107–110, 2016. DOI: 10.1109/ICSESS.2016.7883026.

[10] C. Cornelio, M. Donini, A. Loreggia, M. S. Pini, F. Rossi. Voting with random classifiers (vorace). *arXiv: 1909.08996*, 2019. https://arxiv.org/abs/1909.08996.

[11] X. B. Liu, Z. T. Liu, G. J. Wang, Z. H. Cai, H. Zhang. Ensemble transfer learning algorithm. *IEEE Access*, vol. 6, pp. 2389–2396, 2017. DOI: 10.1109/ACCESS.2017.2782884.

[12] S. J. Winham, R. R. Freimuth, J. M. Biernacka. A weighted random forests approach to improve predictive performance. *Statistical Analysis and Data Mining*, vol. 6, no. 6, pp. 496–505, 2013. DOI: 10.1002/sam.11196.

[13] Y. C. Chen, H. Ahn, J. J. Chen. High-dimensional canonical forest. *Journal of Statistical Computation and Simulation*, vol. 87, no. 5, pp. 845–854, 2017. DOI: 10.1080/00949655.

2016.1231191.

[14] H. F. Zhou, X. Z. Zhao, X. Wang. An effective ensemble pruning algorithm based on frequent patterns. *Knowledge-Based Systems*, vol. 56, pp. 79–85, 2014. DOI: 10.1016/j.knosys.2013.10.024.

[15] Y. Zhang, S. Burer, W. N. Street. Ensemble pruning via semidefinite programming. *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1315–1338, 2006.

[16] L. I. Kuncheva, J. J. Rodríguez. A weighted voting framework for classifiers ensembles. *Knowledge and Information Systems*, vol. 38, no. 2, pp. 259–275, 2014. DOI: 10.1007/s10115-012-0586-6.

[17] A. Kabir, C. Ruiz, S. A. Alvarez. Mixed bagging: a novel ensemble learning framework for supervised classification based on instance hardness. In *Proceedings of IEEE International Conference on Data Mining*, IEEE, Singapore, Singapore, pp. 1073–1078, 2018. DOI: 10.1109/ICDM.2018.00137.

[18] L. V. Utkin, M. S. Kovalev, A. A. Meldo. A deep forest classifier with weights of class probability distribution subsets. *Knowledge-based Systems*, vol. 173, pp. 15–27, 2019. DOI: 10.1016/j.knosys.2019.02.022.

[19] H. Reddy, N. Raj, M. Gala, A. Basava. Text-mining-based fake news detection using ensemble methods. *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 210–221, 2020. DOI: 10.1007/s11633-019-1216-5.

[20] W. G. Yi, J. Duan, M. Y. Lu. Double-layer Bayesian classifier ensembles based on frequent itemsets. *International Journal of Automation and Computing*, vol. 9, no. 2, pp. 215–220, 2012. DOI: 10.1007/s11633-012-0636-2.

[21] G. Wang, J. X. Hao, J. Ma, H. B. Jiang. A comparative assessment of ensemble learning for credit scoring. *Expert Systems with Applications*, vol. 38, no. 1, pp. 223–230, 2011. DOI: 10.1016/j.eswa.2010.06.048.

[22] F. Martínez-Plumed, R. B. Prudêncio, A. Martínez-Usó, J. Hernández-Orallo. Making sense of item response theory in machine learning. In *Proceedings of the 22nd European Conference on Artificial Intelligence*, IOS Press, The Hague, The Netherlands, pp. 1140–1148, 2016. DOI: 10.3233/978-1-61499-672-9-1140.

[23] C. Zanon, C. S. Hutz, H. Yoo, R. K. Hambleton. An application of item response theory to psychological test development. *Psicologia*: *Reflexão e Crítica*, vol. 29, no. 1, Article number 18, 2016. DOI: 10.1186/s41155-016-0040-x.

[24] H. L. Fu, G. Manogaran, K. Wu, M. Cao, S. Jiang, A. M. Yang. Intelligent decision-making of online shopping behavior based on internet of things. *International Journal of Information Management*, vol. 50, pp. 515–525, 2020. DOI: 10.1016/j.ijinfomgt.2019.03.010.

[25] W. R. Gilks, S. Richardson, D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Boca Raton, USA: Chapman & Hall, CRC, 1995.

[26] Y. Chen, T. S. Filho, R. B. C. Prudencio, T. Diethe, P. Flach. β^3-IRT: a new item response model and its applications. *arXiv: 1903.04016*, 2019. https://arxiv.org/abs/1903.04016.

[27] B. W. Junker, R. J. Patz, N. M. VanHoudnos. Markov chain Monte Carlo for item response models. *Handbook of Item Response Theory, Volume Two: Statistical Tools*, W. J. van der Linden, Ed., Boca Raton, USA: Chapman and Hall, CRC, pp. 271–325, 2016.

[28] J. S. Kim, D. M. Bolt. Estimating item response theory models using Markov chain Monte Carlo methods. *Educational Measurement*: *Issues and Practice*, vol. 26, no. 4, pp. 38–51, 2007. DOI: 10.1111/j.1745-3992.2007.00107.x.

[29] M. A. Tanner, W. H. Wong. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 528–540, 1987. DOI: 10.1080/01621459.1987.10478458.

[30] J. H. Albert. Bayesian estimation of normal ogive item response curves using Gibbs sampling. *Journal of Educational Statistics*, vol. 17, no. 3, pp. 251–269, 1992. DOI: 10.3102/10769986017003251.

[31] Y. Y. Sheng. Markov chain Monte Carlo estimation of normal ogive IRT models in matlab. *Journal of Statistical Software*,vol. 25,no. 8,pp. 1–15,2008.DOI:10.18637/jss.v025.i08.

[32] Y. Y. Sheng. Bayesian estimation of the four-parameter IRT model using Gibbs sampling. *International Journal of Quantitative Research in Education*, vol. 2, no. 3–4, pp. 194–212, 2015. DOI: 10.1504/IJQRE.2015.071736.

[33] Y. Noel, B. Dauvier. A beta item response model for continuous bounded responses. *Applied Psychological Measurement*, vol. 31, no. 1, pp. 47–73, 2007. DOI: 10.1177/0146621605287691.

[34] J. C. Xu, Q. W. Ren, Z. Z. Shen. Prediction of the strength of concrete radiation shielding based on LS-SVM. *Annals of Nuclear Energy*, vol. 85, pp. 296–300, 2015. DOI: 10.1016/j.anucene.2015.05.030.

[35] S. Borman. The expectation maximization algorithm: a short tutorial. Submmitted for Publication, vol. 41, 2004.

[36] W. Deng, H. M. Zhao, L. Zou, G. Y. Li, X. H. Yang, D. Q. Wu. A novel collaborative optimization algorithm in solving complex optimization problems. *Soft Computing*, vol. 21, no. 15, pp. 4387–4398, 2017. DOI: 10.1007/s00500-016-2071-8.

[37] M. H. Fang, X. H. Hu, T. T. He, Y. Wang, J. M. Zhao, X. J. Shen, J. Yuan. Prioritizing disease-causing genes based on network diffusion and rank concordance. In *Proceedings of IEEE International Conference on Bioinformatics and Biomedicine*, IEEE, Belfast, UK, pp. 242–247, 2014. DOI: 10.1109/BIBM.2014.6999162.

[38] S. R. Safavian, D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660–674, 1991. DOI: 10.1109/21.97458.

[39] A. Liaw, M. Wiener. Classification and regression by randomforest. *R News*, vol. 2–3, pp. 18–22, 2002.

[40] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002. DOI: 10.1016/S0167-9473(01)00065-2.

[41] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, K. R. Mullers. Fisher discriminant analysis with kernels. In *Proceedings of IEEE Signal Processing Society Workshop*, IEEE, Madison, USA, pp. 41–48, 1999. DOI: 10.1109/NNSP.1999.788121.

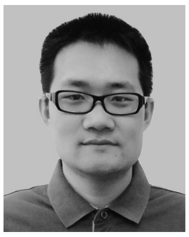
[42] J. A. K. Suykens, J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999. DOI: 10.1023/A:1018628609742.

[43] E. Bauer, R. Kohavi. An empirical comparison of voting classification algorithms: bagging, boosting, and variants.

Machine Learning, vol. 36, no. 1–2, pp. 105–139, 1999. DOI: 10.1023/A:1007515423169.

[44] H. Li, F. D. Chen, K. W. Cheng, Z. Z. Zhao, D. Z. Yang. Prediction of zeta potential of decomposed peat via machine learning: comparative study of support vector machine and artificial neural networks. *International Journal of Electrochemical Science*, vol. 10, no. 8, pp. 6044–6056, 2015.

[45] Y. C. Chen, H. Ha, H. Kim, H. Ahn. Canonical forest. *Computational Statistics*, vol. 29, no. 3–4, pp. 849–867, 2014. DOI: 10.1007/s00180-013-0466-x.



Ziheng Chen received the B. Sc. degree in statistics from Renmin University of China, China in 2016. He is currently a Ph. D. degree candidate in Department of Applied Mathematics and Statistics, Stony Brook University, USA.

His research interests include reinforcement learning, recommending system, tree structure model and ensemble learning theory.

E-mail: ziheng.chen@stonybrook.edu (Corresponding author)

ORCID iD: 0000-0002-2585-637X

Hongshik Ahn received the B. Sc. degree in mathematics from Seoul National University, South Korea, and the Ph. D. degree in statistics from University of Wisconsin-Madison, USA in 1992. From 1992 to 1996, he was a mathematical statistician at the National Center for Toxicological Research, U.S. Food and Drug Administration, and a faculty member in the Department of Applied Mathematics and Statistics at Stony Brook University, USA from 1996 to present. He was the first Vice President of SUNY Korea for two years from 2012. Currently, he is a professor at Stony Brook University. He has published 2 books, 3 book chapters, over 70 papers in peer-reviewed journals, and 25 conference papers.

His research interests include classification of high-dimensional data, tree-structured regression modeling, survival analysis, and multi-step batch testing for infectious diseases.

E-mail: hongshik.ahn@stonybrook.edu