

Discovering Voting Power for Ensemble Methods

Pratik Karmakar^{1,2}[0009–0008–1111–8801], Angelo Saadeh²[0009–0000–2081–6232],
Pierre Senellart^{2,3,4,5}[0000–0002–7909–5369], and
Stéphane Bressan^{*1,4}[0000–0001–5536–3296]

¹ National University of Singapore, Singapore
`pratik.karmakar@u.nus.edu`

² CNRS@CREATE LTD, Singapore
`angelo.saadeh@cnrsatcreate.sg`

³ DI ENS, ENS, PSL University, CNRS, Inria, Paris, France
`pierre@senellart.com`

⁴ IPAL, CNRS, Singapore

⁵ Institut Universitaire de France

Abstract. Ensemble methods aggregate the predictions of multiple models by some form of weighted voting. In this work, we consider the impact of the choice of the assignment of voting power to every individual model on the performance of ensemble methods. We empirically and comparatively evaluate the accuracy and running time of the different power voting ensemble methods using standard classifiers and mainstream classification benchmarks. The results show that power ensemble voting outperforms the equal-power baseline, and that unsupervised learning of the voting power can be competitive with respect to supervised learning; within supervised approaches, learning voting power through Shapley values and regression outperforms simply using accuracy.

Keywords: Ensemble Methods · Voting Power · Shapley Values · Inverse Entropy · Truth Discovery · Classification

1 Introduction

Ensemble methods aggregate the predictions of multiple models, to sometimes obtain better performance than any single model [29]. The four main classes of ensemble methods are *boosting* [11], *bagging* [2], *stacking* [32] and *voting*. In this work, we focus on *voting* because it is the setting in which models are trained independently. In addition, we also focus on classification tasks.

Most existing ensemble voting methods assign equal voting power to each model [25]. But treating all models equally may yield less accurate results since expert and non-expert models are treated equally. For this reason, some works have considered assigning a different voting power to each model based on some criterion, allowing models with a higher voting power to influence the final prediction more. The voting power of a model is a weight that is used for weighted voting. This usually outperforms equal voting power, as shown in [17,20].

* Stéphane Bressan tragically passed away recently, after contributing to this work.

Most works use the models’ accuracy or a regression on their predictions to assign voting powers. These two techniques require an auxiliary labelled dataset. Therefore, we refer to them as *supervised*. Our work presents other ways to assign voting powers to the models. The first is to use the game-theoretic solution known as *Shapley values* [28]. A Shapley value is a number computed using a set function; in our example, it would indicate the influence of a model on the accuracy of the aggregation of predictions. In general, Shapley values take exponential time to compute. A more computationally efficient but similar algorithm is *Leave-One-Out* [14], which has polynomial-time complexity. Whether regression, accuracy, or Shapley values are used to compute voting powers, these supervised methods require additional information other than the predictions: a ground truth to compare to the models’ predictions. Therefore, the next technique we use to compute the voting powers is *truth discovery* (or truth finding) [33,12], which does not require any other information except the models’ predictions. An example state-of-the-art truth discovery algorithm is CRH [18], which assigns weights to the models by minimizing some objective function, these weights serving as voting powers. Finally, we introduce using the *inverse of the entropy* of each prediction vector, based on a vector of weights assigned to each class by each model and used to compute how confident the models are about their predictions. Note that Shapley values and inverse entropy for ensemble voting are original contributions.

In ensemble voting for classification tasks, the set of candidates are target classification classes, the models’ predictions serve as votes, and the set of voters is the set of models. For example, the set of candidates in a logistic regression in the MNIST data set for the optical character recognition of digits [7] is $\mathcal{C} = \{0, 1, \dots, 9\}$. In voting theory, voting amounts to ordering the set of candidates according to the voter’s preferences. Votes are aggregated using a voting mechanism such as plurality voting or Borda count [1]. More precisely, in a plurality voting mechanism, the candidate who gets the most first-choice votes is the winner. However, in Borda count, each candidate is given some points based on rank of preference. The Borda winner is the candidate who gets the most points once the points across all the voters are summed up. In our work, in addition to plurality voting, we use a customized version of the Borda voting because the output prediction of a model naturally assigns a weight to each class (candidate). Although other voting mechanisms could be used in ensemble voting, these two are the most commonly used for their simplicity and accuracy [31].

The paper is as follows: After presenting related work in Section 2, we define and explain the different methods we use to compute and assign voting powers in Section 3. In Section 4 we empirically evaluate the accuracy and running time of the methods using standard classifiers and mainstream classification benchmarks in the areas of computer vision and Web classification. The code required to repeat our experiments is available at https://github.com/pratik2358/voting_power_ensemble_anon.

2 Related Work

Ensemble methods aim at better predictive performance than the one obtained from one model [9]. In some ensemble techniques, like boosting, models are trained sequentially. Each model’s training depends on the performance of the previous model. An example of a Boosting algorithm is AdaBoost [11], which has many variants [27]. Bagging [2], unlike boosting, does not require interactions between models. However, bagging involves training multiple models on different subsets of the training data using the same training algorithm. Then predictions from these models are aggregated using averaging for regressions and majority voting for classification tasks. Bagging is different from voting in that voting does not necessarily rely on the same type of model. Another primary ensemble technique is stacking [32], where predictions from multiple models are aggregated using weights learned by regressing the predictions to a target prediction. Our work focuses on ensemble *voting* where the models need not be trained using the same algorithm or on the same data distribution.

Different voting mechanisms can be used in ensemble voting for classification tasks. The most common one is plurality voting where the winner is the candidate that gets the most vote. It is also loosely referred to as majority voting (which *stricto sensu* is when the candidate gets more than 50% of the votes). Plurality and majority voting are used for most ensemble voting [25]. Examples of other voting algorithms are Condorcet [3], instant run-off method, and Borda count [1]. In Borda count, each candidate is assigned some points, then the candidate that gets the most points is the winner. Borda count is also used in ensemble voting [22]. The authors of [31] compare voting mechanisms in ensemble voting, and they conclude that plurality voting and Borda count have the lowest complexity-to-performance ratio.

Previously discussed ensemble voting techniques assign equal voting power to the models. However, there are ways to assign different voting powers to models. The first approach is to use the accuracy of each model as its voting power. For instance, [13] uses the accuracy of a model as its voting power. Another strategy is to include only models that meet a minimum threshold accuracy and then use it as their voting power [23]. In other approaches also based on accuracy [26,8], the models that correctly make predictions incorrectly classified by most other classifiers are given more voting power. In other words, the focus is more on true positives than overall accuracy. The second approach consists of assigning voting powers as the weights learned from a regression of the predictions to fit a target prediction. The authors of [19] do so using a linear regression model. Other works [10] use a custom loss function that assigns voting powers based on the reliability of each model for specific output classes. These two approaches require an auxiliary labelled dataset, which we call supervised. Truth discovery is also used to assign voting powers [16] in an unsupervised manner. In summary, existing works mainly use regression or the accuracy of models to assign a voting power for each model. Our work defines other techniques to discover and assign voting powers. Moreover, we compare them by evaluating their accuracy and running time.

Table 1. Classification of power finding methods.

Prediction format	Unsupervised	Supervised
Vector format	Inverse entropy	Regression
Label format	Truth discovery (CRH)	Accuracy Shapley values Leave-one-out (LOO)

3 Methodology

In our work, we assign voting powers to models and use plurality and Borda voting for ensemble voting for an ℓ -class classification problem (for simplicity, we write the classes as $0, \dots, \ell-1$). To compute and assign the voting powers, we use supervised power learning mechanisms like accuracy, regression, and Shapley values, as well as unsupervised mechanisms like truth discovery and inverse entropy. These mechanisms can be further categorized with respect to the required format of the predictions, which can either be *soft* classifications, where the output of the classifier is a vector of ℓ scores assigned to all classes, or *hard* predictions of the most common label. For example, for a “cat” (0) or “dog” (1) binary classification task, a vector output would be $(0.8, 0.2)$, and a label output would be $0 = \arg \max_{0,1}(0.8, 0.2)$. The different power-assigning techniques we consider are summarized in Table 1. We write $M(q) \in \{0, \dots, \ell-1\}$ for the hard prediction of model M for class q and $\mathbf{M}(q) \in \mathbb{R}_+^\ell$ for the soft vector of scores.

Supervised methods require a labelled set of queries \mathcal{F} to query the models and have their predictions compared with a ground truth. Unsupervised methods do not require such an auxiliary labelled dataset – but the computation of voting power for unsupervised method uses an auxiliary set of queries, with no access to ground truth, also called \mathcal{F} here.

We note that in the case where a Borda voting mechanism is used to aggregate the predictions after the voting power computation, then choosing a method from Table 1 that requires a vector as a voting format would not require more information than the methods that require only the label as a prediction; this is because the Borda voting mechanism requires a vector format anyway.

As previously mentioned, for all methods, we assume access to a certain number of queries, which are used as auxiliary information to compute the model voting powers, and we call this set of m queries \mathcal{F} (for supervised methods, we need access to the ground-truth label of each query in \mathcal{F}). If we have n different models M_1, M_2, \dots, M_n , the goal is to obtain the voting powers $\mathbf{w} = (w_1^*, w_2^*, \dots, w_n^*)$ which represent the voting power of each model. The voting powers need not sum up to 1; the equal-power case is where all w_i^* are identical.

3.1 Voting Mechanisms

Plurality voting. In plurality voting (with voting power), we assume each classifier M_i outputs its prediction label $M_i(q)$ for query q . The winning prediction

for query q is the one which maximizes the amounts of votes, weighted by their voting power: $\arg \max_{0 \leq k < \ell} \sum_{i=1}^n w_i^* \delta_{k, M_i(q)}$ with ties randomly resolved and $\delta_{ij} := 1$ if $i = j$, and 0 otherwise.

Borda voting. In Borda voting (with voting power), each classifier M_i outputs a vector $\mathbf{M}_i(q)$ of scores for each label, for query q . The winning prediction for query q is the one which maximizes the total score, weighted by the voting power of each model: $\arg \max_{0 \leq k < \ell} \sum_{i=1}^n w_i^* (\mathbf{M}_i(q))_k$.

3.2 Supervised Methods

For supervised methods, we assume the ground truth $\gamma(q)$ of every query $q \in \mathcal{F}$.

Accuracy. Accuracy of the models is a naïve choice for the model voting powers. We compute the accuracy of each model M_i on \mathcal{F} as: $w_i^* := \frac{1}{|\mathcal{F}|} \sum_{q \in \mathcal{F}} \delta_{M_i(q), \gamma(q)}$.

Regression. In both Plurality and Borda settings, it is not desired to make the mechanisms perform optimally only for one query. We are rather interested in optimizing the performance of all \mathcal{F} . For any q , let $\mathbf{M}(q)$ be the $n \times \ell$ matrix: $\mathbf{M}(q) := (\mathbf{M}_1(q)^T \cdots \mathbf{M}_n(q)^T)$. We aim to minimize the empirical risk by choosing as voting power: $\mathbf{w}^* := \arg \min_{\mathbf{w}} \mathbb{E}_{q \sim \mathcal{F}} [\|(w \times \mathbf{M}(q)) - \gamma(q)\|_2^2]$, where $\gamma(q)$ is represented as a one-hot vector. This, now, is a convex optimization (regression) problem that we solve using gradient descent. In our case, having access to the set of queries \mathcal{F} , we minimize the empirical risk of \mathcal{F} , assuming i.i.d.

Shapley values. Shapley values [28] are used to assess the contribution of a party (or model in our case) to a specific task. Computing the Shapley value requires us to define the task through a set function called *value function* given by $v : 2^{\{1, \dots, n\}} \rightarrow \mathbb{R}$. We take the value function v to be the accuracy of equal power plurality voting on \mathcal{F} : $v := S \mapsto \frac{1}{|\mathcal{F}|} \sum_{q \in \mathcal{F}} \delta_{\gamma(q), \arg \max_k \sum_{i \in S} \delta_{k, M_i(q)}}$. The Shapley value of model M_i is: $w_i^* := \frac{1}{n!} \sum_{S \subseteq \{1, \dots, n\} \setminus \{i\}} |S|!(n - |S| - 1)! (v(S \cup i) - v(S))$ which, for each combination of models, measures the impact of model M_i . Shapley values are chosen as a power-assigning technique, for it satisfies four significant axioms: Efficiency, Symmetry, Dummy, and Additivity [21], which makes it an attribution method for fair reward distribution. However, their exponential computational complexity limits the use of Shapley values.

Leave-One-Out (LOO) The size of the powerset used to compute Shapley values increases exponentially with the number of models, making the Shapley values impossible to compute. A more computationally efficient similar algorithm is leave-one-out [4,14], which has polynomial time complexity. The leave-one-out value for a model M_i is given by: $\tilde{w}_i^* := v(\{1, \dots, n\} \cup i) - v(\{1, \dots, n\})$ with v defined as for Shapley values. While not as well-grounded as Shapley values for power assignment, this method is much less resource-intensive, making it a more practically feasible attribution method for some instances.

3.3 Unsupervised Methods

We now assume a set \mathcal{F} of queries, but no knowledge of the ground truth.

Inverse Entropy. The entropy of a vector defining a discrete probability distribution $P = (p_0, p_1, \dots, p_{\ell-1})$ is defined as $H(P) = -\sum_{k=0}^{\ell-1} p_k \log(p_k)$. Then, the inverse entropy for model M_i is defined as $w_i^* := \frac{|\mathcal{F}|}{\sum_{q \in \mathcal{F}} H(M_i(q))}$. The rationale behind using the inverse mean entropy on \mathcal{F} as the model’s weight is quite straightforward: entropy is a measure of uncertainty of a model’s decision. The higher the model’s uncertainty, the higher the entropy value. Thus, inverse entropy assigns a higher weight to the models, which are more certain about their decisions on \mathcal{F} . Note that if a model is fully certain, i.e., if $H(M_i(q))$ is 0 for all q , then the inverse entropy is $+\infty$ for this model, meaning this model’s predictions will be selected as the ensemble one; this extreme case does not occur in practice with the classifiers we considered in our experiments.

CRH weights. We use the CRH [18] truth discovery algorithm to assign voting power, which attempts to optimize voting power w_i^* and predictions (*truth values*) y_q for each query $q \in \mathcal{F}$. This is done iteratively: first, y_q are set to the equal-power plurality vote; then, voting powers are first estimated by minimizing the following loss function: $\mathbf{w}^* := \arg \min_{\mathbf{w}} \sum_{i=1}^n w_i \sum_{q \in \mathcal{F}} \delta(y_q, M_i(q))$, with constraint $\sum_i \exp(-w_i) = 1$. Then, the y_q votes are re-estimated using plurality vote with voting power \mathbf{w}^* ; new values for \mathbf{w}^* are then iteratively computed. This is repeated until some convergence criterion (see [18]) is met. The optimization is solved using Lagrange multipliers [18].

4 Performance Evaluation

Experimental Setup. We performed our experiments using four different standard datasets, with a variety of classifiers trained over these: MNIST [7] handwritten digits, CINIC-10 [5] image classification, DMOZ URL Classification⁶, and Webpage Phishing Detection [15].

We need models of varying accuracy to evaluate different power-assigning methods in our setting. We assess model quality by evaluating accuracy on a test set distinct from the training set. To simulate models with varying levels of quality, we employ two strategies. In *Label Flipping*, the training dataset is partitioned into subsets where labels are flipped with varying probabilities; more precisely, for a specific model M_i , a probability $p_i \in [0, 1]$ is drawn and M_i is then trained on a dataset where a fraction p_i of the labels has been flipped. In *Class Imbalance*, the training dataset is partitioned into subsets with varying degrees of class imbalance; models trained on these subsets demonstrate differing accuracy based on the extent of class imbalance.

⁶ <https://www.kaggle.com/datasets/shawon10/url-classification-dataset-dmoz>

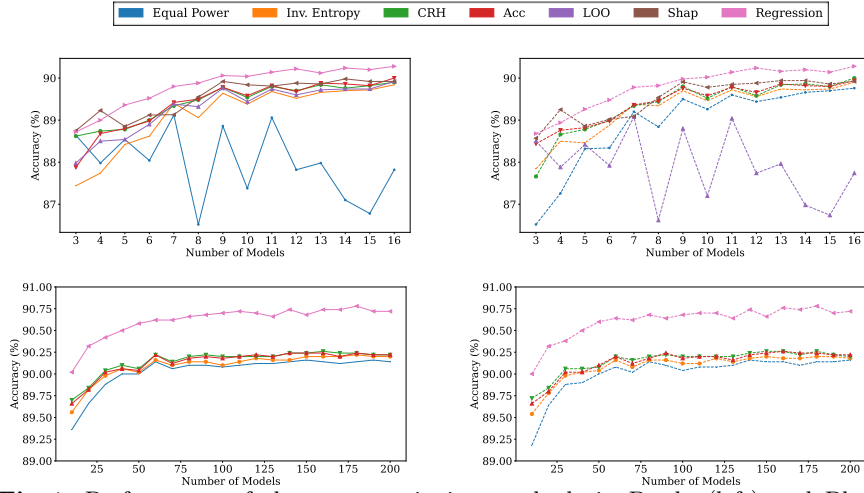


Fig. 1. Performance of the power-assigning methods in Borda (left) and Plurality (right) settings for up to 16 models (top) and up to 200 models (bottom) (MNIST data).

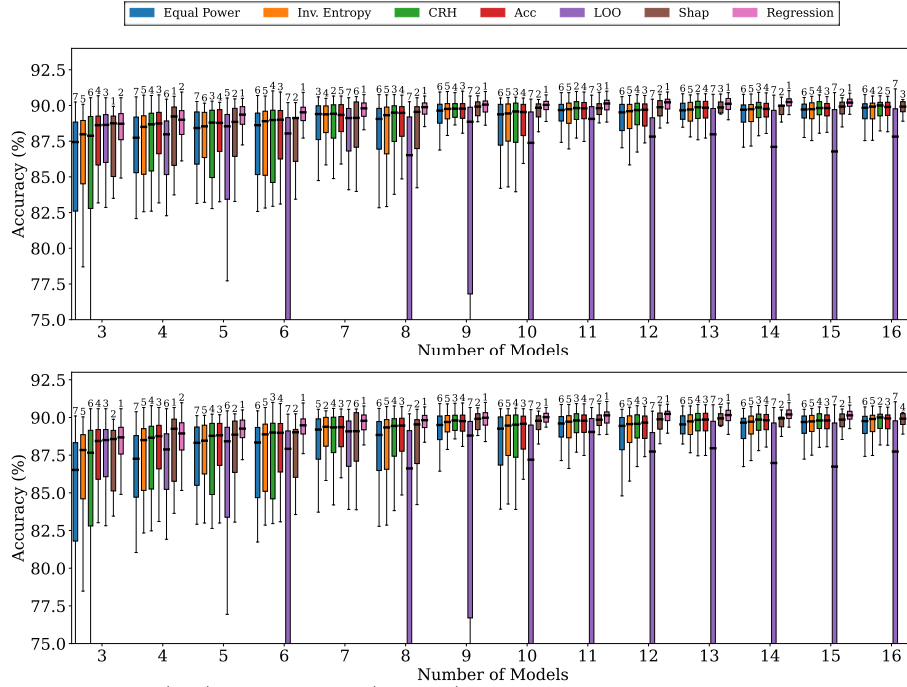


Fig. 2. Borda (top) and Plurality (bottom) voting with different voting power assigning methods (MNIST data).

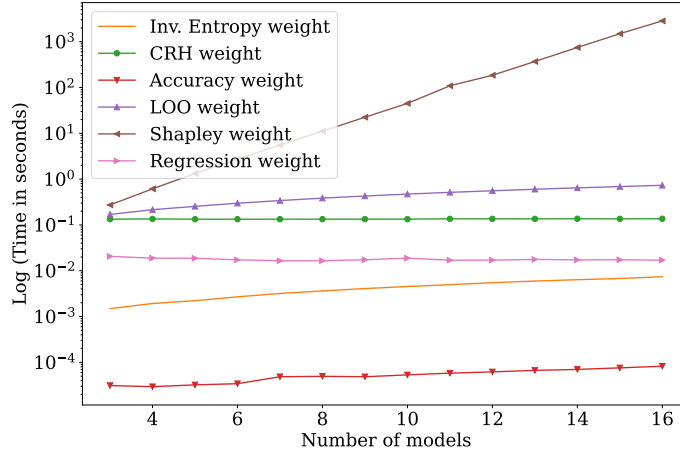


Fig. 3. Time comparison of the power-assigning methods (MNIST data). The y-axis uses a logarithmic time scale.

Every dataset is split into a training set (used to train the classifiers, possibly on different subsets), a validation set (playing the role of the set \mathcal{F} used for computing the voting power for every power-assigning technique), and a test set (used to evaluate the overall performance of the ensemble methods). To ensure statistically significant results, the experiments on MNIST data are conducted 75 times, and experiments on other datasets are conducted 20 times.

MNIST Dataset. Our experiments use a logistic regression classifier, trained using PyTorch [24]. We randomly pick 5000 samples from the MNIST dataset to train each model. These random subsets are not necessarily disjoint. We do this to keep the number of training data points constant throughout all the experiments’ models. We simulate varying model quality using label-flipping noise in the training subsets. As expected, higher label flipping results in lower model accuracy.

We vary the number of models. To better evaluate the voting power computation techniques, we repeat the experiments multiple times with a different training dataset for each model, for each number of models to be aggregated. More precisely, we use fifteen different label-flipping probability assignments and each label flipping is used in five different experiments. For each number of models to be aggregated, we thus repeat the experiment seventy-five times on different training data.

In Figures 1 and 2, we present the performance of different methods used for Borda and Plurality voting, respectively, with varying numbers (3–16) of models to aggregate. The ranks of the methods for every set of model numbers are indicated above the boxes in Figure 2. In Figure 1, we also present the experimental results for up to 200 models (bottom). In the case of 200 models,

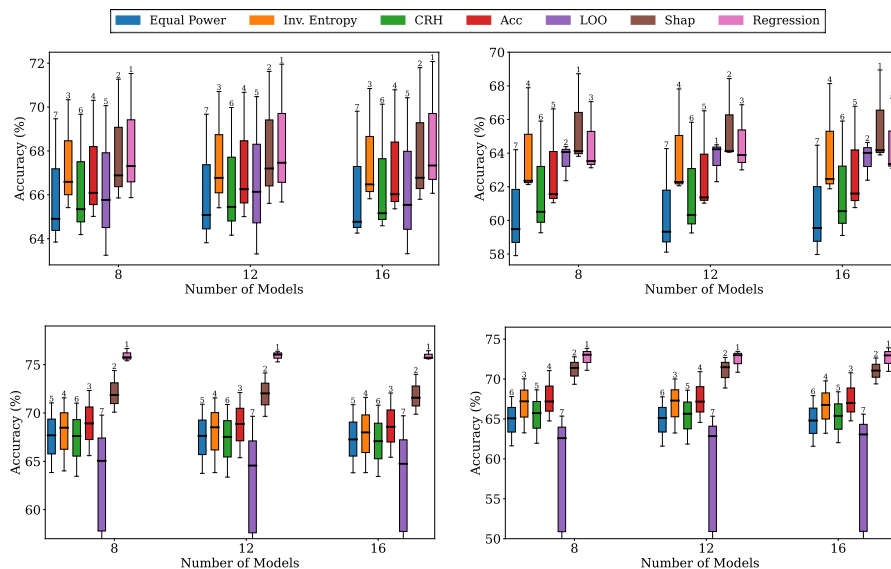


Fig. 4. Performance of the power-assigning methods in Borda (left) and Plurality (right) settings for label-flipping noise (top) and class-imbalance noise (bottom) (CINIC-10 data)

we do not show the results of using Shapley values as voting powers due to the exponential computational cost, leading to resource constraints. Figure 2 compares the variance of the voting power assigning methods and the effect of the number of models on the variance of aggregate decisions. In Figure 3, we compare the time complexities (in log-scale) of finding the voting powers using different methods. The equal-power method is not presented here as we do not compute any voting power in this case, and thus, it is the least costly method.

Our empirical results on this dataset indicate that the voting powers found using regression perform better than the others among the chosen methods of voting power assignment. Shapley’s value as voting power outperforms the rest of the methods in most cases. Among the unsupervised methods, CRH performs better than the inverse entropy method, showing a higher median and lower variance in performance in most cases, but it is of higher time complexity. The performance of LOO is poor and exhibits a high variance.

CINIC-10 Dataset. The CINIC-10 dataset consists of 60,000 images from CIFAR-10 and 210,000 images from ImageNet [6], all categorized using the same classes as CIFAR-10. We have used models of two different architectures here: a Convolutional Neural Network (CNN) with three convolutional layers and VGG16 [30]. We train $2n$ CNN models and $2n$ VGG16 models. Specifically, n models are trained on n subsets of the CIFAR-10 training subset of CINIC-10, and the other n models are trained on n subsets of the ImageNet training subset of CINIC-10. These $4n$ models, exhibiting different qualities, are then used for our

experiments. In our experiments, $n \in \{2, 3, 4\}$. The validation and test splits consist of images from CIFAR-10 and ImageNet, ensuring a comprehensive model performance evaluation. We apply label flipping and class imbalance separately to create distinct models for separate experiments. We use this strategy to split the dataset so that our models of different architectures learn from two different distributions and are then tested on a subset containing data from both.

We present the empirical results in Figure 4. In most cases, the ensembles using voting powers outperform the equal-power ensemble. Powers found using regression and Shapley values perform the best throughout. While in the case of label-flipping noise, the equal-power ensemble performs the worst, in class imbalance, voting powers found using LOO perform the worst.

DMOZ URL Classification Dataset. The URL classification dataset from DMOZ used in our study contains URLs and corresponding classes in the DMOZ catalogue for 1,562,978 web pages spanning 15 categories. We split the data into disjoint partitions: 70% for training, 15% for testing, and 15% for validation. We use Multinomial Naive Bayes (MNB) models. To train each model, we randomly sample 90% of the training data and introduce varying levels of label-flipping noise. This process ensures that resulting models exhibit different levels of accuracy. Our experiments are conducted with ensembles of 3, 5, 7, and 10 models.

As shown in Figure 5, voting powers derived from Shapley values outperform all other methods. Conversely, powers determined using LOO perform the worst. Notably, all other power voting methods surpass the equal-power voting approach. We also notice that the inverse entropy method does not perform significantly better than equal-power voting.

Webpage Phishing Detection Dataset. This dataset comprises data from 11,430 web pages, each characterized by 88 features, including the URL, URL length, hostname length, domain age, Google index, and page rank, among others. The data is divided into disjoint partitions for training (60%), validation (20%), and testing (20%). We use a 3-layer neural network for this experiment. We divide the training partition into n disjoint subsets and then add varying amounts of label-flipping noise to the subsets to train models of different quality. In our experiments $n \in \{5, 10, 15\}$.

The empirical results are shown in Figure 6. All voting power methods outperform the equal-power approach except the inverse entropy method, particularly in the Borda voting scenario. Shapley values as voting powers consistently outperform all other methods across Borda and Plurality voting strategies.

Other voting mechanisms. Six techniques were employed to compute and assign voting powers to sources, which were then utilized in power voting. In addition to Plurality and Borda, we evaluated other voting mechanisms, too.

First is Condorcet voting, in which a voter ranks their vote from highest preference to lowest preference, and a candidate is iteratively eliminated until a majority winner prevails. This means that a candidate is the most preferred candidate of half of the voters. We tested this voting mechanism, which performed

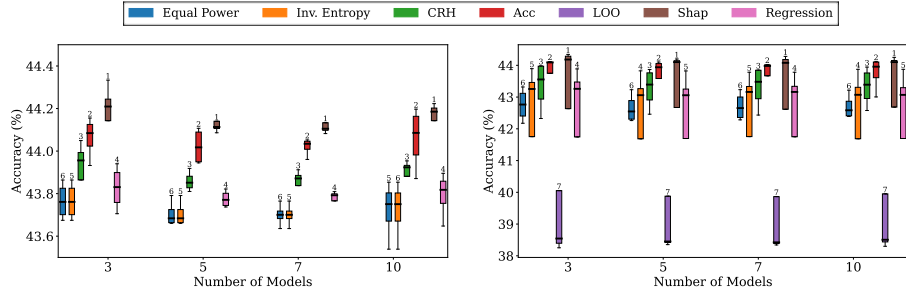


Fig. 5. Performance of the power-assigning methods in Borda (left) and Plurality (right) settings for label-flipping noise (DMOZ data)

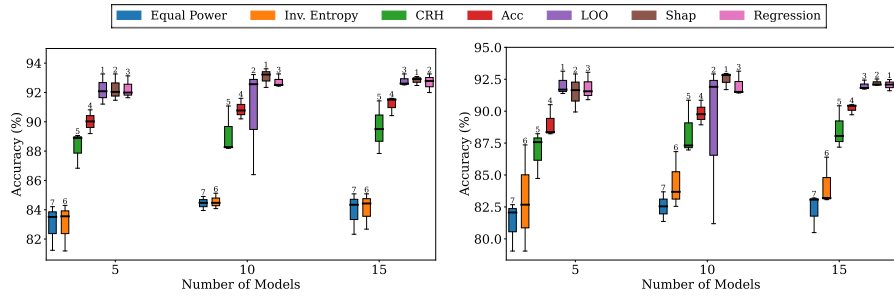


Fig. 6. Performance of the power-assigning methods in Borda (left) and Plurality (right) settings for label-flipping noise (Phishing data)

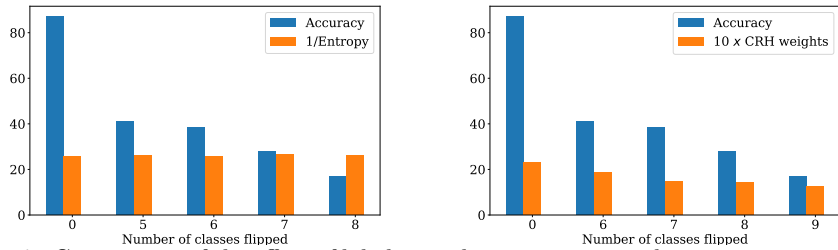


Fig. 7. Comparison of the effect of label-interchange in training data on inverse entropy weights (left) and CRH weights (right) (MNIST data)

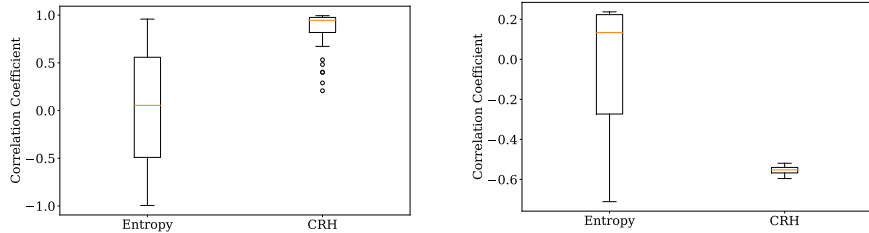


Fig. 8. Correlation between model accuracy and model voting powers computed using inverse entropy and CRH when the complete label interchange pattern is random (left) or deterministic (right) across different training datasets (MNIST data)

badly but have not reported its results. Indeed, there is a loss of granularity when a vector of votes is transformed from probability of preference (like in Borda) to ranked candidates with no probabilities or values assigned to the candidates. The lost information is crucial for the voters because they might have a first-choice candidate with 100% confidence in Borda and all other candidates with a value of zero. Second, we tried taking into account the vote of the voter with the highest voting power and ignoring the other voters, which we do not report here. The tests also showed a low performance for the method, which shows that a collective decision would be better than an individual one. More particularly, taking into account the vote of the model with the best accuracy is not better than taking the average prediction of all the models. On one hand, this could partially be explained by the fact that more models considered means more data was used in the training process, yielding a better generalization. On the other hand, there is a loss of diversity in opinion, i.e., some voters may train their models differently, making their insight important for the aggregation result. In summary, the choice between Borda and plurality voting does not significantly affect the aggregation result, as seen in the experiments. However, choosing aggregate votes using Condorcet voting or choosing the “best” voter’s predictions drastically reduces accuracy.

Voting power computation and assignment. It was shown that most voting power computation techniques give similar results. The difference lies in the information required to compute these powers. Indeed, using truth discovery (CRH) or the entropy techniques yields almost the same results as using Shapley or the local accuracy of the models. However, computing the latter two powers requires access to labelled data to assess the models.

The name truth discovery is misleading because “truth” is defined as what the majority thinks on average. Therefore, if most models are wrong on average, then truth discovery will not yield good results. Usually, this is an improbable scenario, and even in this case, the other techniques will not be better. Furthermore, while truth discovery only uses prediction from the sources, entropy requires having the prediction as a probability vector, which is more information than a simple label for truth discovery. Nevertheless, if the aggregator would like to use Borda voting, then the aggregator would need the full vector of probabilities anyway. Therefore, entropy would not require more information than truth discovery.

The inverse entropy method has a lower computational cost than the CRH method, as seen in Figure 3, but we may have situations of trade-off between the two. We explain one instance partly explaining the performance gap between the two methods. Using inverse entropy can go severely wrong as it depends on the models’ confidence. To illustrate this phenomenon, we train five logistic regression models on five subsets of the MNIST handwritten digits dataset. The first subset has all its true labels intact. From the second subset onward, we randomly interchange class labels of 5, 6, 7, and 8 classes. In Figure 7, we see the decrease in accuracy when models trained on these datasets are tested on a subset disjoint from these training subsets. Inverse entropy values remain the same for

all models. Even though models trained on interchanged class labels have learned the wrong labels, the learning degree is not affected by risk minimization on the training sets.

The similar quality disorder in training data does not affect the CRH method as much as it affects the inverse entropy method. In Figure 7, we present one instance of the setup mentioned above to compare the effect of the data quality flaw mentioned above in assigning voting power using the inverse entropy method (left) and CRH method (right). In Figure 8 (left), we show the correlation coefficient between the accuracy of the models trained on data with 0, 5, 6, 7, and 8 randomly interchanged labels and the voting powers were computed using inverse entropy and CRH (20 runs). We notice a significantly higher correlation between the voting powers and the accuracy of the models in the case of CRH than in the case of inverse entropy. The robustness of CRH lies in the fact that to find the power of a model, this method has to consider votes from other models in the ensemble, too. Thus, if all the models do not deviate from the truth, there’s still a probability of the truth found using CRH converging to the ground truth. Thus, for CRH to fail deterministically, the label interchange must be deterministic, not random. However, this is a very unlikely phenomenon as there is rarely a chance that all the models are being trained on different datasets where the interchange of labels has taken place similarly across all training datasets. We demonstrate this in Figure 8 (right).

Results show that using regression voting powers yields the best accuracy. Among the other informed methods, Shapley’s voting powers outperform other methods. However, this technique is very resource-intensive, and its computation complexity increases with the number of voters. Even though there are better approximations than the one we use, LOO, they are also computationally expensive. LOO performs very poorly, especially when the number of models is high because it assesses the added value of a model on all the other models. In plurality and Borda voting, the probability that one model changes the aggregation result becomes very low as the number of models grows.

5 Conclusion and Future Work

This work extends ensemble voting techniques by exploring multiple ways of assigning voting powers to individual classifiers (including original approaches such as inverse entropy and Shapley values) paired with voting mechanisms like plurality and Borda. The results confirm that assigning voting powers to classifiers outperforms the equal-power baseline. Within supervised power-assigning methods, the classical accuracy-based voting powers are less effective than regression and Shapley values. Unsupervised power-assigning methods often perform competitively with supervised ones without a labelled ground truth.

Moving forward, we are applying this research to knowledge transfer frameworks that aggregate predictions via plurality voting from multiple trained models to label an unlabelled dataset that is then used to train a new privacy-preserving model. The obtained model would not reveal information on the data

used because calibrated noise is added during the aggregation process. Since our techniques could potentially be applied at the aggregation step of this scenario, our future work consists of doing a privacy analysis on our aggregation techniques by computing their sensitivity, adding noise accordingly, and evaluating how each method affects the privacy and performance of the new model.

Acknowledgments. This research is part of the program DesCartes and is supported by the National Research Foundation, Prime Minister’s Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program. This work is made in memory of our co-author, colleague, and friend Stéphane Bressan, who unexpectedly passed away in December 2024.

References

1. de Borda, J.C.: Mémoire sur les élections au scrutin. *Histoire de l’Académie Royale des Sciences* (1781)
2. Breiman, L.: Bagging predictors. *Mach. Learn.* **24**(2), 123–140 (1996)
3. de Condorcet, N.: Essai sur l’application de l’analyse à la probabilité des décisions rendues à la pluralité des voix. Cambridge University Press (2014)
4. Cook, R.D.: Detection of influential observation in linear regression. *Technometrics* **42**(1), 65–68 (2000)
5. Darlow, L.N., Crowley, E.J., Antoniou, A., Storkey, A.J.: Cinic-10 is not imagenet or cifar-10. arXiv preprint arXiv:1810.03505 (2018)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
7. Deng, L.: The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.* **29**(6), 141–142 (2012)
8. Dogan, A., Birant, D.: A weighted majority voting ensemble approach for classification. In: 2019 4th International Conference on Computer Science and Engineering (UBMK). pp. 1–6 (2019)
9. Dong, X., Yu, Z., Cao, W., Shi, Y., Ma, Q.: A survey on ensemble learning. *Frontiers Comput. Sci.* **14**(2), 241–258 (2020)
10. Ekbal, A., Saha, S.: Weighted vote-based classifier ensemble for named entity recognition: A genetic algorithm-based approach. *ACM Trans. Asian Lang. Inf. Process.* **10**(2), 9:1–9:37 (2011)
11. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Saitta, L. (ed.) *Machine Learning, Proceedings of the Thirteenth International Conference (ICML ’96)*, Bari, Italy, July 3-6, 1996. pp. 148–156. Morgan Kaufmann (1996)
12. Galland, A., Abiteboul, S., Marian, A., Senellart, P.: Corroborating information from disagreeing views. In: *Proceedings of the third ACM international conference on Web search and data mining*. pp. 131–140 (2010)
13. Georgiou, H.V., Mavroforakis, M.E.: A game-theoretic framework for classifier ensembles using weighted majority voting with local accuracy estimates. *CoRR abs/1302.0540* (2013)
14. Ghorbani, A., Zou, J.: Data shapley: Equitable valuation of data for machine learning. In: *International conference on machine learning*. pp. 2242–2251. PMLR (2019)
15. Hannousse, A., Yahiouche, S.: Web page phishing detection (2021), mendeley Data, V3

16. Jin, Y., Yang, Z., He, Y., Bao, X., Wu, G.: Ensemble classification method based on truth discovery. In: 2019 IEEE International Conference on Big Knowledge (ICBK). pp. 122–128. IEEE (2019)
17. Kuncheva, L.I., Rodríguez, J.J.: A weighted voting framework for classifiers ensembles. *Knowl. Inf. Syst.* **38**(2), 259–275 (2014)
18. Li, Y., Li, Q., Gao, J., Su, L., Zhao, B., Fan, W., Han, J.: Conflicts to harmony: A framework for resolving conflicts in heterogeneous data by truth discovery. *IEEE Trans. Knowl. Data Eng.* **28**(8), 1986–1999 (2016)
19. Li, Y., Luo, Y.: Performance-weighted-voting model: an ensemble machine learning method for cancer type classification using whole-exome sequencing mutation. *Quant. Biol.* **8**(4), 347–358 (2020)
20. Mahendran, N., Vincent, P.M.D.R., Srinivasan, K., Chang, C., Garg, A., Gao, L., Gutiérrez-Reina, D.: Sensor-assisted weighted average ensemble model for detecting major depressive disorder. *Sensors* **19**(22), 4822 (2019)
21. Molnar, C., Casalicchio, G., Bischl, B.: *Interpretable Machine Learning - A Brief History, State-of-the-Art and Challenges*, Communications in Computer and Information Science, vol. 1323. Springer (2020)
22. Niyas K. P., M., Paramasivan, T.: Improving Alzheimer’s classification using a modified borda count voting method on dynamic ensemble classifiers. *Knowl. Inf. Syst.* **66**(8), 4755–4787 (2024)
23. Okuboyejo, D.A., Olugbara, O.O.: Classification of skin lesions using weighted majority voting ensemble deep learning. *Algorithms* **15**(12), 443 (2022)
24. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. *NIPS 2017 Autodiff Workshop* (2017)
25. Raza, K.: Improving the prediction accuracy of heart disease with ensemble learning and majority voting rule. In: *U-Healthcare Monitoring Systems*, pp. 179–196. Elsevier (2019)
26. Rojarath, A., Songpan, W.: Cost-sensitive probability for weighted voting in an ensemble model for multi-class classification problems. *Appl. Intell.* **51**(7), 4908–4932 (2021)
27. Shahraki, A., Abbasi, M., Haugen, Ø.: Boosting algorithms for network intrusion detection: A comparative evaluation of real adaboost, gentle adaboost and modest adaboost. *Eng. Appl. Artif. Intell.* **94**, 103770 (2020)
28. Shapley, L.S.: A value for n-person games. In: Kuhn, H.W., Tucker, A.W. (eds.) *Contributions to the Theory of Games II*, pp. 307–317. Princeton University Press, Princeton (1953)
29. Shwartz-Ziv, R., Armon, A.: Tabular data: Deep learning is not all you need. *Inf. Fusion* **81**, 84–90 (2022)
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014)
31. Torres-Sospedra, J., Hernandez-Espinosa, C., Fernandez-Redondo, M.: A comparison of combination methods for ensembles of RBF networks. In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* vol. 2, pp. 1137–1141. IEEE (2005)
32. Wolpert, D.H.: Stacked generalization. *Neural Networks* **5**(2), 241–259 (1992)
33. Yin, X., Han, J., Yu, P.S.: Truth discovery with multiple conflicting information providers on the web. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining.* pp. 1048–1052 (2007)