Granul. Comput. (2017) 2:131–139
DOI 10.1007/s41066-016-0034-1

ORIGINAL PAPER

Granular computing-based approach for classification towards reduction of bias in ensemble learning

Han Liu[1] · Mihaela Cocea[1]

Received: 25 August 2016 / Accepted: 2 November 2016 / Published online: 11 November 2016

Abstract Machine learning has become a powerful approach in practical applications, such as decision making, sentiment analysis and ontology engineering. To improve the overall performance in machine learning tasks, ensemble learning has become increasingly popular by combining different learning algorithms or models. Popular approaches of ensemble learning include Bagging and Boosting, which involve voting towards the final classification. The voting in both Bagging and Boosting could result in incorrect classification due to the bias in the way voting takes place. To reduce the bias in voting, this paper proposes a probabilistic approach of voting in the context of granular computing towards improvement of overall accuracy of classification. An experimental study is reported to validate the proposed approach of voting using 15 data sets from the UCI repository. The results show that probabilistic voting is effective in increasing the accuracy through reduction of the bias in voting. This paper contributes to the theoretical and empirical analysis of causes of bias in voting, towards advancing ensemble learning approaches through the use of probabilistic voting.

Keywords Granular computing · Machine learning · Ensemble learning · Bagging · Boosting · Probabilistic voting

✉ Han Liu
han.liu@port.ac.uk

Mihaela Cocea
mihaela.cocea@port.ac.uk

[1] School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace, Portsmouth PO1 3HE, UK

1 Introduction

Machine learning has become an increasingly powerful approach in real applications, such as decision making (Das et al. 2016; Xu and Wang 2016), sentiment analysis (Liu 2012; Pedrycz and Chen 2016) and ontology engineering (Pedrycz and Chen 2016; Roussey et al. 2011). In practice, machine learning can be involved in classification and regression, which are considered as supervised learning tasks. In other words, training data used in classification and regression are labelled. Also, machine learning can be involved in association and clustering, which are considered as unsupervised learning tasks. In other words, training data used in association and clustering are unlabelled. This paper focuses on classification tasks.

In the context of classification, popular machine learning methods include decision tree learning (Quinlan 1986; Chen et al. 2016), rule learning (Liu and Gegov 2016b; Du et al. 2011; Rodrguez-Fdez et al. 2016), Bayesian learning (Zhang et al. 2009; Yager 2006), instance-based learning (Tu et al. 2016; Gonzlez et al. 2016; Langone and Suykens 2017) and perceptron learning (Shi et al. 2016; da Silva and de Oliveira 2016). Both decision tree learning and rule learning aim to learn a set of rules. The difference between these two types of learning is that the former is aimed at learning of rules in the form of a decision tree, e.g. ID3 (Quinlan 1986) and C4.5 (Quinlan 1993), whereas the latter aims to learn if-then rules directly from training instances, e.g. Prism (Cendrowska 1987) and IEBRG (Liu and Gegov 2016a). In particular, decision tree learning typically follows the divide and conquer approach, whereas rule learning mainly follows the separate and conquer approach. Bayesian learning works based on the assumption that all the input attributes are totally independent of each other, e.g. Naive Bayes (Barber 2012). In this context, each attribute–value pair would be independently correlated to each of the possible classes, which means that a posterior probability is provided between the attribute–value pair and the class. Instance-based learning generally involves predicting test instances on the basis of their similarity to the training instances, e.g. K nearest neighbour (Liu et al. 2016a). In other words, this type of learning does not involve learning models in the training stage, but just aims to classify each instance to the category to which the majority of its nearest neighbours (the instances most similar to it) belong. Perceptron learning aims to build a neural network topology that consists of a number of layers, each of which has a number of nodes and represents a perceptron. Some popular methods of neural network learning include backpropagation and probabilistic neural networks (Kononenko and Kukar 2007).

In general, each machine learning algorithm has its advantages and disadvantages. To improve the overall accuracy of classification, ensemble learning has been adopted. Popular ensemble learning approaches include Bagging (Breiman 1996) and Boosting (Freund and Schapire 1996). Both approaches involve voting in the testing stage towards the final classification. In particular, Bagging employs majority voting (Kononenko and Kukar 2007; Li and Wong 2004) by means of selecting the class with the highest frequency towards classifying an unseen instance and Boosting employs weighted voting (Kononenko and Kukar 2007; Li and Wong 2004) by means of selecting the class with the highest weight for the same purpose. Both majority voting and weighted voting are considered to be biased to always select the class with the highest frequency or weight, which may result in overfitting of training data (Barber 2012). The aim of this paper is to provide theoretical and empirical analysis of bias in voting and contribute towards reduction of the bias in voting through the use of granular computing concepts. In particular, the probabilistic voting approach, which has been proposed in Liu et al. (2016a) for advancing individual learning algorithms that involve voting, is used in this paper to advance ensemble learning approaches. More details on the probabilistic voting are presented in Sect. 3. How this voting approach is linked to granular computing concepts is also justified in Sect. 3.

The rest of this paper is organised as follows: Sect. 2 presents ensemble learning concepts and the two popular approaches namely Bagging and Boosting; Sect. 3 presents a probabilistic approach of voting in the context of granular computing and argues that this approach can effectively reduce the bias in voting towards classifying an unseen instance; Sect. 4 reports an experimental study to validate the proposed approach of voting, and the results are also discussed to show the extent to which the accuracy of classification is improved through the reduction of the bias in voting. Section 5 summarises the contributions of this paper and suggests further directions for this research area towards further advances in ensemble learning.

2 Related work

This section describes in depth the concepts of ensemble learning and reviews two popular approaches, namely Bagging and Boosting. It also highlights how the voting involved in these two approaches can lead to incorrect classification.

2.1 Ensemble learning concepts

The concepts of Ensemble learning are usually used to improve overall accuracy, i.e. in order to overcome the limitations that each single learning algorithm has its own disadvantages and the quality of original data may not be good enough. In particular, this purpose can be achieved through scaling up algorithms (Kononenko and Kukar 2007) or scaling down data (Kononenko and Kukar 2007). The former means a combination of different learning algorithms which are complementary to each other. The latter means pre-processing of data towards the improvement of data quality. In practice, ensemble learning can be done both in parallel and sequentially.

In the context of classification, the parallel ensemble learning approach works by combining different learning algorithms, each of which generates a model independently on the same training set. In this way, the predictions of the models learned by these algorithms are combined toward classifying unseen instances. This way belongs to scaling up algorithms because different algorithms are combined to generate a stronger hypothesis. In addition, the parallel ensemble learning approach can also be achieved using a single learning algorithm to generate models independently on different sample sets of training instances. In this context, the sample set of training instances can be provided by horizontally selecting the instances with replacement or vertically selecting the attributes without replacement. This way belongs to scaling down data, because it is aimed to pre-process data towards reducing the variability of the data, leading to the reduction of the variance in classification results.

In the sequential ensemble learning approach, accuracy can also be improved through scaling up algorithms or scaling down data. In the former way, different algorithms are combined in such a way that the first algorithm learns to build a model and then the second algorithm learns to correct the model and so on. In this way, there are no changes made to the training data. In the latter way, in contrast, the same algorithm is used iteratively on different versions of the training data. At each iteration, there is a model learned, which is then evaluated on the basis of the validation data. According to the estimated quality of the model, the training instances are weighted to different extents and then used for the next iteration. In the testing stage, these models learned at different iterations make predictions independently and their predictions are then combined towards classifying unseen instances.

For both parallel and sequential ensemble learning approaches, voting is involved in the testing stage to combine the independent predictions of different learning algorithms or models towards classifying an unseen instance. Some popular methods of voting include majority voting and weighted voting. As mentioned in Sect. 1, the former one is typically used for the Bagging approach and the latter is typically used for the Boosting approach (Kononenko and Kukar 2007; Li and Wong 2004). More details on these approaches of ensemble learning and voting are presented in the following subsections.

2.2 Bagging

The term Bagging stands for bootstrap aggregating. It is a popular method developed by Breiman (1996) and follows the parallel ensemble learning approach. Bagging involves sampling of data with replacement. In particular, the Bagging method typically takes n samples, with each sample of size m, where m is the size of the training set, in which the instances from the training set are randomly selected into each sample set. This indicates that some instances in the training set may appear more than once in one sample set and some other instances may never appear in that sample set. On average, a sample is expected to contain 63.2% of the training instances (Kononenko and Kukar 2007; Li and Wong 2004). In the training stage, the classifiers, each resulting from a particular sample set mentioned above, are parallel to each other. In the testing stage, their independent predictions are combined towards predicting the final classification through majority voting (also known as equal voting).

The detailed procedure of Bagging is illustrated in Fig. 1. As concluded in Kononenko and Kukar (2007), Li and Wong (2004), Bagging is robust and does not lead to overfitting due to the increase of the number of generated models. Therefore, it is useful especially for those non-stable learning methods with high variance. A popular example of Bagging is Random Forests (Breiman 2001), which is illustrated in Fig. 2.

Although the Bagging approach has the advantages mentioned above, it still has bias in voting, which may result in incorrect classification. In particular, the majority voting involved in Bagging works based on the assumption that the training data is complete and, thus, the class most frequently predicted by base classifiers is the most accurate one. However, it is fairly difficult to guarantee that the above assumption is reliable. Section 3 will present how this problem can be addressed using probabilistic voting.

2.3 Boosting

Boosting follows the sequential learning approach, which is introduced in Freund and Schapire (1996), Kononenko and Kukar (2007), Li and Wong (2004). In other words, the generation of a single classifier depends on the experience gained from its former classifier (Li and Wong 2004). Each single classifier is assigned a weight depending on its accuracy estimated using validation data. The stopping criteria are satisfied while the error is equal to 0 or greater than 0.5 (Li and Wong 2004). In the testing stage, each single classifier makes an independent prediction in a similar way to Bagging, but the final prediction is made based on weighted voting among these independent predictions.

As concluded in Kononenko and Kukar (2007), Boosting frequently outperforms Bagging, and can also be applied with those stable learning algorithms with low variance in addition to unstable ones, in contrast to Bagging. However, Boosting may generate an ensemble learner that overfits training data. In this case, the performance of the ensemble learner is worse than that of a single learner. A popular example of Boosting is referred to as Adaboost, which is illustrated below (Freund and Schapire 1996):

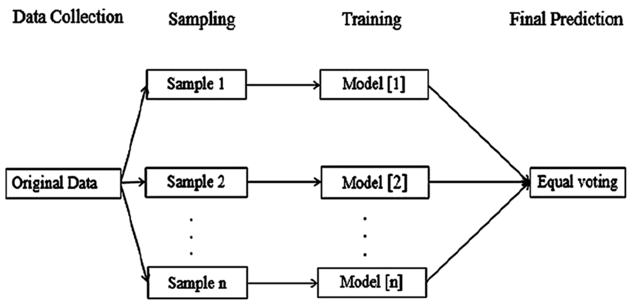

Fig. 1 Bagging approach (Liu et al. 2016c)

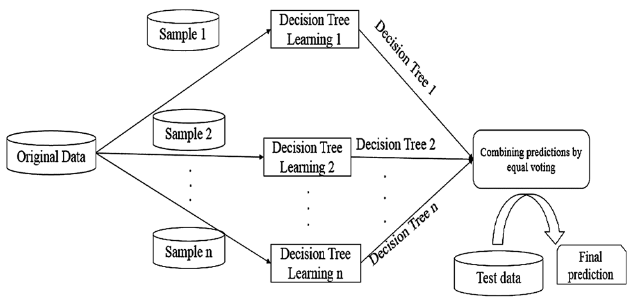

Fig. 2 Random forests (Liu et al. 2016c)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialise $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error $\epsilon_t = Pr_i[h_t(x_i) \neq y_i]$.
- Choose $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{Bmatrix} e^{-\alpha_t}, & \mathrm{if h_t(x_i) = y_i} \\ e^{\alpha_t}, & \mathrm{if h_t(x_i) \neq y_i} \end{Bmatrix} = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalisation factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis: $H(x) = \mathrm{sign}(\sum^T \alpha_t h_t(x))$

In the above illustration, x_i indicates an input vector and y_i indicates the class label assigned to x_i, where i is the index of an instance. Also, X and Y represent the domain and range of the given data set, respectively. In addition, the distribution D_t reflects how each instance is weighted at each particular iteration of the procedure for the Adaboost. The symbol t represents the number of the current iteration and α_t represents the weight of the classifier learned at the iteration t.

However, similar to Bagging, Boosting also has bias in voting although it has the advantages mentioned above. In particular, the weighted voting involved in Boosting also works based on the assumption that the training data are complete and, thus, the most highly weighted class is the most accurate one. However, it is fairly difficult to guarantee that the above assumption is reliable. This problem can be addressed using the probabilistic voting and the details are given in Sect. 3.

3 Granular computing-based approach for voting

As pointed out in Sects. 2.2 and 2.3, both majority voting and weighted voting are considered to be biased leading to incorrect classifications. In other words, the above two types of voting can be seen as deterministic voting, since they both work in the context of deterministic logic by assuming that there is no uncertainty for classifying an unseen instance. This section describes the concepts of granular computing including the link to probabilistic logic, and then proposes to use probabilistic voting as a granular computing-based approach towards reduction of the bias in voting. The significance of the probabilistic voting is also outlined by analysing the advantages of granular computing.

3.1 Overview of granular computing

Granular computing is an emerging approach of information processing. It is applied with two main aims as stressed in Yao (2005). One aim is to adopt structured thinking at the philosophical level and the other one is to conduct structured problem solving at the practical level. As described in Hu and Shi (2009), granular computing generally involves decomposition of a whole into several parts. In practice, this means to decompose a complex problem into several simpler sub-problems.

The fundamentals of granular computing generally involve information granulation which includes probabilistic sets, fuzzy sets and rough sets. Deterministic sets can be seen as special cases of all the three above types of sets. In particular, a probabilistic set is judged as a deterministic set while each element has a 100% chance to belong to the set. Also, a fuzzy set is judged as a deterministic set while each element has a full membership to the set, i.e. the fuzzy membership degree is 100%. Similarly, a rough set is judged as a deterministic set while each element unconditionally belongs to the set. The above description indicates that deterministic sets are used in the context of deterministic logic, whereas the other three types of sets are used in the context of non-deterministic logic.

In the context of probabilistic sets, as described in Liu et al. (2016b), each set employs a chance space which can be partitioned into a number of subspaces. Each of these subspaces can be viewed as a granule that can be randomly selected towards enabling an event to occur. In this context, all these granules make up the whole chance space. As also described in Liu et al. (2016b), each element in a probabilistic set is granted a probability towards getting a full membership to the set. In the context of granular computing, the probability can be viewed as a percentage of the granules that make up the chance space. For example, if an element is given a probability of 80% towards getting a full membership to a set, then the element would be assigned 80% of the granules that enable the full membership to be granted.

In the context of fuzzy sets, as described in Liu et al. (2016b), each element in a fuzzy set has a certain degree of membership to the set, i.e. an element belongs to a set to a certain extent. In the context of granular computing, the membership can be partitioned into a number of parts. Each part of the membership can be viewed as a granule. For example, if an element is given a membership degree of 80% to a set, then the element would be assigned 80% of the granules that certainly belong to the set. This is very similar to the example that a club offers different levels of memberships, which provides the members with different levels of access to the resources and the facilities.

In the context of rough sets, as described in Liu et al. (2016b), each rough set employs a boundary region to allow some elements to be restored conditionally on the basis of the insufficient information. In other words, all these elements within the boundary region are only given conditional memberships to the set. This is because these elements have only partially met the conditions towards being members of the set. Once the conditions have been fully met, these elements would be granted full memberships to the set. In the context of granular computing, the condition for an element to belong to the set can be partitioned into a number of subconditions. Each of these subconditions can be viewed as a granule. As defined in Liu et al. (2016b), possibility is aimed to measure the extent to which the condition is met. For example, if the possibility that an element belongs to a set is 80%, then the element would be assigned 80% of the granules, each of which leads to the partial fulfilment towards getting the membership.

In practice, the concepts of granular computing have been applied broadly in many areas such as artificial intelligence (Wilke and Portmann 2016; Skowron et al. 2016), computational intelligence (Dubois and Prade 2016; Kreinovich 2016; Livi and Sadeghian 2016), and machine learning (Min and Xu 2016; Peters and Weber 2016; Antonelli et al. 2016). In addition, ensemble learning is also considered as a granular computing approach since it involves decomposition of a data set into a number of overlapping samples in the training stage and a combination of predictions by different classifiers towards classifying a test instance. The similar perspective has also been pointed out in Hu and Shi (2009). The next section presents in detail how the concepts of granular computing can be used towards reduction of bias in voting in the context of ensemble learning.

3.2 Probabilistic voting

Probabilistic voting (Liu et al. 2016a) is considered to be inspired by nature and biology in the context of granular computing, since the voting is made on the basis of the hypothesis that the class with the highest frequency or weight only has the best chance of being selected towards classifying an unseen instance. In other words, it is not guaranteed that the class with the highest frequency or weight will definitely be selected to be assigned to the unseen instance. In this paper, probabilistic voting is used for both Bagging and Boosting towards improving the overall classification accuracy. In particular, majority voting (involved in Bagging) and weighted voting (involved in Boosting) are both replaced with probabilistic voting. The procedure of probabilistic voting is illustrated below:

Step 1: calculating the weight W_i for each single class i.
Step 2: calculating the total weight W over all classes.
Step 3: calculating the percentage of weight P_i for each single class i, i.e. $P_i = W_i \div W$.
Step 4: Randomly selecting a single class i with the probability P_i towards classifying an unseen instance.

The following example relating to Bayes Theorem is used for the illustration of the above procedure:

Inputs (binary): x_1, x_2, x_3
Output (binary): y
Probabilistic correlation:

$$P(y=0|x_1=0)=0.4, P(y=1|x_1=0)=0.6,$$
$$P(y=0|x_1=1)=0.5, P(y=1|x_1=1)=0.5;$$
$$P(y=0|x_2=0)=0.7, P(y=1|x_2=0)=0.3,$$
$$P(y=0|x_2=1)=0.6, P(y=1|x_2=1)=0.4;$$
$$P(y=0|x_3=0)=0.5, P(y=1|x_3=0)=0.5,$$
$$P(y=0|x_3=1)=0.8, P(y=1|x_3=1)=0.2;$$

While $x_1=0, x_2=1, x_3=1, y=?$

Following Step 1, the weight W_i for each single value of y is:

$$\begin{aligned}P(y=0|x_1=0,x_2=1,x_3=1) &= P(y=0|x_1=0)\\ &\times P(y=0|x_2=1)\\ &\times P(y=0|x_3=1) = 0.4\times 0.6\times 0.8 = 0.192\\ P(y=1|x_1=0,x_2=1,x_3=1) &= P(y=1|x_1=0)\\ &\times P(y=1|x_2=1)\\ &\times P(y=1|x_3=1) = 0.6\times 0.4\times 0.2 = 0.048\end{aligned}$$

Following Step 2, the total weight W is $0.24 = 0.192 + 0.048$.

Following Step 3, the percentage of weight P_i for each single value of y is:

Percentage for $y=0$: $P_0 = 0.192 \div 0.24 = 80\%$
Percentage for $y=1$: $P_1 = 0.048 \div 0.24 = 20\%$

Following Step 4, $y=0$ (80% chance) or $y=1$ (20% chance).

In the above illustration, both majority voting and weighted voting would result in 0 being assigned to y due to its higher frequency or weight shown in Step 4. In particular, in the context of majority voting, Step 4 would indicate that the frequency for y to equal 0 is 80% and the one for y to equal 1 is 20%. Also, in the context of weighted voting, Step 4 would indicate that over the total weight the percentage of the weight for y to equal 0 is 80% and the percentage of the weight for y to equal 1 is 20%. Therefore, both types of voting would choose to assign y the value of 0. However, in the context of probabilistic voting, Step 4 would indicate that y could be assigned either 0 (with 80% chance) or 1 (with 20% chance). In this way, the bias in voting can be effectively reduced towards improvement of overall accuracy of classification in ensemble learning.

The probabilistic voting approach illustrated above is very similar to natural selection which is one step of the procedure of genetic algorithms (Man et al. 1996), i.e. the probability of a class being selected is very similar to the fitness of an individual involved in natural selection. In particular, the way of selecting a class involved in Step 4 of the above procedure is inspired by the Roulette Wheel Selection (Lipowski and Lipowska 2012).

In the context of granular computing, the frequency of a class can be viewed as an information granule that enables the class to be selected for being assigned to a test instance. Similarly, the weight of a class can be viewed as a part of information granules that enable the class to be selected towards classifying an unseen instance. From this point of view, the class with the highest frequency of being predicted by base classifiers means to have been assigned the most information granules that enable the class to be selected for being assigned to a test instance. Similarly, the class with the highest weight means to have been assigned the highest percentage of the information granules that enable the class to be selected towards classifying an unseen instance. More details on information granules can be found in Pedrycz and Chen (2011, 2015a, b).

As mentioned in Sect. 1, for classifying test instances, the Bagging method is biased to always select the most frequently occurring class and the Boosting method is biased to always select the most highly weighted class. This is due to the assumption that all the independent predictions by the base classifiers provide a complete and highly trusted set of information granules, each of which votes towards one class and against all the other classes. However, it is fairly difficult to guarantee that a set of granules is complete, due to the fact that the training and validation sets are very likely to be incomplete in practice. Also, it is commonly known that a training set may be imbalanced, due to insufficient collection of data, which is likely to result in a class being assigned much more information granules than the other classes. In addition, a learning algorithm may not be suitable to learn a model on a particular sample set. In this case, the information granules, which are provided from the predictions by the models learned by that algorithm, would be much less trusted.

In the context of machine learning, it has been argued in Liu et al. (2016a) that voting based on heuristics such as frequency or weight is biased. In particular, as mentioned in Sect. 1, the probabilistic voting approach has been applied to two popular single learning algorithms (Naive Bayes and K Nearest Neighbour) for reduction of bias in voting and the experimental results were encouraging. Since this type of voting is involved in ensemble learning as well, probabilistic voting could also lead to improved results in this context.

In ensemble learning, Bagging needs to draw a number of samples of the original data on a random basis and Boosting needs to iteratively evaluates the weight of training instances. The nature of the Bagging method may result in poor samples of training data being drawn in terms of incompleteness and imbalance. The nature of the Boosting method may result in poor evaluation of training instances in terms of their weights. If the employed learning algorithms are not suitable to the sampled data for Bagging or the weighted data for Boosting, then the frequency or the weight of classes would be much less trusted for classifying test instances. Therefore, the majority voting involved in Bagging and the weighted voting involved in Boosting are considered to be biased. This is very similar to the human reasoning approach that people generally make decisions and judgements based on their previous experience without the guarantee that the decisions and judgements are absolutely right (Liu et al. 2016a). However, the frequency or the weight of a class can fairly be used to reflect the chance of the class being selected towards classifying a test instance, especially when the above conjecture concerning low-quality training data cannot be proved in a reasonable way. The impact of probabilistic voting on the Bagging and Boosting approaches is investigated experimentally in the following section.

4 Experimental results

The probabilistic voting illustrated in Sect. 3.2 is validated in an experimental study to investigate its impact on the Bagging and Boosting approaches, by comparing the results with traditional Bagging (with majority voting) and Boosting (with weighted voting) in terms of classification accuracy. In particular, the Random Forests and Adaboost methods are used for this experimental study due to the fact that they are the popular examples of Bagging and Boosting, respectively, in practical applications.

The experiments are conducted on 15 data sets retrieved from the UCI repository (Lichman 2013). The characteristics of these data sets are shown in Table 1. In general, all the chosen data sets have lower dimensionality (less than 100) and smaller number of instances (less than 1000) except for the hypothyroid data set. The choice of these data sets was made on the basis of the computational complexity of the two ensemble learning methods used, namely Random Forests and Adaboost. In particular, as mentioned in Sect. 2.2, the Bagging approach needs to

Table 1 Data sets

Name	Attribute types	Attributes	Instances	Classes
breast-cancer	Discrete	9	286	2
breast-w	Continuous	10	699	2
ecoli	Continuous	8	336	8
glass	Continuous	10	214	6
haberman	Mixed	4	306	2
heart-c	Mixed	76	920	4
heart-h	Mixed	76	920	4
heart-statlog	Continuous	13	270	2
hypothyroid	Mixed	30	3772	4
ionosphere	Continuous	34	351	2
iris	Continuous	5	150	3
labor	Mixed	17	57	2
sonar	Continuous	61	208	4
vote	Discrete	17	435	2
wine	Continuous	14	178	3

draw *n* samples with the same size of *m*, where *m* is the size of the training set. Therefore, the computational complexity of the Random Forests can be considered to be *n* times the complexity of a single learning algorithm such as decision tree learning, if no parallelisation is adopted. The same also applies to Adaboost, especially when considering that the nature of the Boosting approach is not parallelism. In addition, these data sets contain both discrete and continuous attributes as shown in Table 1. This is to investigate the impact of probabilistic voting in the case of both types of attributes.

The experiments are conducted by splitting a data set into a training set and a test set in the ratio of 70:30. For each data set, the experiment is repeated 10 times and the average of the accuracies is taken for comparative validation. As mentioned above, due to the higher computational complexity of ensemble learning approaches, cross validation (Kononenko and Kukar 2007) is not used in this experimental study. The results are shown in Table 2.

In Table 2, the second and third columns (Random Forest I and II) indicate the results for Random Forests with majority voting (Breiman 2001) and Random Forests with probabilistic voting (proposed in Sect. 3.2), respectively. Similarly, the fourth and fifth columns (Adaboost I and II) indicate the results for Adaboost with weighted voting (Freund and Schapire 1996) and Adaboost with probabilistic voting (proposed in Sect. 3.2), respectively.

The results show that, except for the hypothyroid data set, probabilistic voting can help both Random Forest and Adaboost to effectively improve the overall accuracy of classification. Regarding the case on the hypothyroid data set, it is the only data set that has a large number of

Table 2 Classification accuracy

Data set	Random forest I (%)	Random forest II (%)	Adaboost I (%)	Adaboost II (%)
breast-cancer	70	78	74	77
breast-w	95	96	94	96
ecoli	83	85	65	68
glass	69	80	45	52
haberman	68	74	72	78
heart-c	78	81	82	84
heart-h	82	87	79	80
heart-statlog	77	84	82	88
hypothyroid	98	98	95	95
ionosphere	89	94	89	90
iris	94	96	93	97
labor	90	95	91	95
sonar	76	83	75	83
vote	95	97	95	98
wine	94	98	88	91

instances (higher than 1000) as shown in Table 1, but the classification accuracy stays the same when using probabilistic voting for both Random Forest and Adaboost. A possible explanation regarding this phenomenon may be that larger data would usually be of higher completeness and the classification result is, thus, less impacted by the bias in voting compared with the use of smaller data sets. A similar point has been given in Liu et al. (2016a) in terms of the likelihood of overfitting.

In addition, while the UCI data sets are a good benchmark for judging new approaches, they are known to be cleaner (i.e. contain fewer errors in the data) and more complete than data used in real-life applications, especially when considering the current vast volumes of data and the need to analyse data streams. Consequently, the benefits of probabilistic voting could be higher on this type of data where the assumptions of completeness and sample representativeness are rarely met; however, further experimentation is required to assess the benefits of probabilistic voting in this context.

5 Conclusion

In this paper, we have discussed in the context of granular computing how the current deterministic ways of voting in ensemble learning methods are biased through the assumptions of completeness of data and sample representativeness, which are rarely met, especially in the context of big data.

To address this issue, we proposed probabilistic voting, which is conceptually close to the idea of natural selection in genetic algorithms. To validate the proposed voting approach and its impact on classification accuracy, we experimented with 15 UCI data sets and two popular ensemble approaches, i.e. Bagging and Boosting. More specifically, the Random Forest and Adaboost algorithms were used. In both cases, the results show an increase in accuracy with probabilistic voting.

In this paper, we addressed the bias involved in the testing stage, i.e. in voting. However, as argued in Liu and Gegov (2015), it is also significant to effectively employ learning algorithms that are combined on a competitive basis and used in the training stage of ensemble learning, towards improvements in the overall accuracy of classification. From this point of view, a further direction for this work is to propose a probabilistic approach, similar to probabilistic voting, towards natural selection of learning algorithms on the basis of their fitness, and to investigate further how this probabilistic approach impacts on the performance of ensemble learning.

Acknowledgments The authors acknowledge support for the research reported in this paper through the Research Development Fund at the University of Portsmouth.

References

Antonelli M, Ducange P, Lazzerini B, Marcelloni F (2016) Multi-objective evolutionary design of granular rule-based classifiers. Granul Comput 1(1):37–58

Barber D (2012) Bayesian reasoning and machine learning. Cambridge University Press, Cambridge

Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Cendrowska J (1987) Prism: an algorithm for inducing modular rules. Int J Man-Mach Stud 27:349–370

Chen Y-L, Wu C-C, Tang K (2016) Time-constrained cost-sensitive decision tree induction. Inf Sci 354:140–152

da Silva AJ, de Oliveira WR (2016) Comments on quantum artificial neural networks with applications. Inf Sci 370:120–122

Das S, Kar S, Pal T (2016) Robust decision making using intuitionistic fuzzy numbers. Granul Comput 1:1–14

Du Y, Hu Q, Zhu P, Ma P (2011) Rule learning for classification based on neighborhood covering reduction. Inf Sci 181(24):5457–5467

Dubois D, Prade H (2016) Bridging gaps between several forms of granular computing. Granul Comput 1(2):115–126

Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: Machine learning: proceedings of the 13th international conference. Bari, Italy, pp 148–156

Gonzlez M, Bergmeir C, Triguero I, Rodrguez Y, Bentez JM (2016) On the stopping criteria for k-nearest neighbor in positive unlabeled time series classification problems. Inf Sci 328:42–59

Hu H, Shi Z (2009) Machine learning as granular computing. In: IEEE international conference on granular computing. Nanchang, Beijing, pp 229–234

Kononenko I, Kukar M (2007) Machine learning and data mining: introduction to principles and algorithms. Horwood Publishing Limited, Chichester

Kreinovich V (2016) Solving equations (and systems of equations) under uncertainty: how different practical problems lead to different mathematical and computational formulations. Granul Comput 1(3):171–179

Langone R, Suykens JA (2017) Supervised aggregated feature learning for multiple instance classification. Inf Sci 375(1):234–245

Li J, Wong L (2004) Rule-based data mining methods for classification problems in biomedical domains. In: A tutorial note for the 15th European conference on machine learning (ECML) and the 8th European conference on principles and practice for knowledge discovery in databases (PKDD), Pisa, Italy

Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml

Lipowski A, Lipowska D (2012) Roulette-wheel selection via stochastic acceptance. Phys A Stat Mech Appl 391(6):2193–2196

Liu B (2012) Sentiment analysis and opinion mining. Morgan and Claypool Publishers, San Rafael

Liu H, Gegov A (2015) Collaborative decision making by ensemble rule based classification systems. Springer, Switzerland, pp 245–264

Liu H, Gegov A (2016a) Induction of modular classification rules by information entropy based rule generation. Springer, Switzerland, pp 217–230

Liu H, Gegov A (2016b) Rule based systems and networks: deterministic and fuzzy approaches. In: IEEE international conference on intelligent systems. Sofia, Bulgaria, pp 316–321

Liu H, Gegov A, Cocea M (2016a) Nature and biology inspired approach of classification towards reduction of bias in machine learning. In: International conference on machine learning and cybernetics. Jeju Island, South Korea, pp 588–593

Liu H, Gegov A, Cocea M (2016b) Rule based systems: a granular computing perspective. Granul Comput 1(4):259–274

Liu H, Gegov A, Cocea M (2016c) Rule based systems for big data: a machine learning approach. Springer, Switzerland

Livi L, Sadeghian A (2016) Granular computing, computational intelligence, and the analysis of non-geometric input spaces. Granul Comput 1(1):13–20

Man KF, Tang KS, Kwong S (1996) Genetic algorithms: concepts and applications. IEEE Trans Ind Electron 43(5):519–534

Min F, Xu J (2016) Semi-greedy heuristics for feature selection with test cost constraints. Granul Comput 1(3):199–211

Pedrycz W, Chen S-M (2011) Granular computing and intelligent systems: design with information granules of higher order and higher type. Springer, Heidelberg

Pedrycz W, Chen S-M (2015a) Granular computing and decision-making: interactive and iterative approaches. Springer, Heidelberg

Pedrycz W, Chen S-M (2015b) Information granularity, big data, and computational intelligence. Springer, Heidelberg

Pedrycz W, Chen S-M (2016) Sentiment analysis and ontology engineering: an environment of computational intelligence. Springer, Heidelberg

Peters G, Weber R (2016) Dcc: a framework for dynamic granular clustering. Granul Comput 1(1):1–11

Quinlan JR (1986) Induction of decision trees. Mach Learn 1:81–106

Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc, San Francisco

Rodrguez-Fdez I, Mucientes M, Bugarn A (2016) Fruler: fuzzy rule learning through evolution for regression. Inf Sci 354:1–18

Roussey C, Pinet F, Kang MA, Corcho O (2011) An introduction to ontologies and ontology engineering. Springer, London, pp 9–38

Shi K, Liu X, Tang Y, Zhu H, Zhong S (2016) Some novel approaches on state estimation of delayed neural networks. Inf Sci 372:313–331

Skowron A, Jankowski A, Dutta S (2016) Interactive granular computing. Granul Comput 1(2):95–113

Tu E, Zhang Y, Zhu L, Yang J, Kasabov N (2016) A graph-based semi-supervised k nearest-neighbor method for nonlinear manifold distributed data classification. Inf Sci 367:673–688

Wilke G, Portmann E (2016) Granular computing as a basis of human data interaction: a cognitive cities use case. Granul Comput 1(3):181–197

Xu Z, Wang H (2016) Managing multi-granularity linguistic information in qualitative group decision making: an overview. Granul Comput 1(1):21–35

Yager RR (2006) An extension of the naive bayesian classifier. Inf Sci 176(5):577–588

Yao Y (2005) Perspectives of granular computing. In: Proceedings of 2005 IEEE international conference on granular computing. Beijing, China, pp 85–90

Zhang M-L, Pea JM, Robles V (2009) Feature selection for multi-label naive bayes classification. Inf Sci 179(19):3218–3229