Inner Ensembles: Using Ensemble Methods Inside the Learning Algorithm

Houman Abbasian[1], Chris Drummond[2], Nathalie Japkowicz[1], and Stan Matwin[3,4]

[1] School of Electrical Engineering and Computer Science
University of Ottawa,
Ottawa, Ontario,Canada, K1N 6N5
habba057@uottawa.ca, nat@site.uottawa.ca
[2] National Research Council of Canada,
Ottawa, Ontario, Canada, K1A 0R6
Christopher.Drummond@nrc-cnrc.gc.ca
[3] Dalhousie University, Halifax, Canada
stan@cs.dal.ca
[4] Institute for Computer Science, Polish Academy of Sciences, Poland

Abstract. Ensemble Methods represent an important research area within machine learning. Here, we argue that the use of such methods can be generalized and applied in many more situations than they have been previously. Instead of using them only to combine the output of an algorithm, we can apply them to the decisions made inside the learning algorithm, itself. We call this approach Inner Ensembles. The main contribution of this work is to demonstrate how broadly this idea can applied. Specifically, we show that the idea can be applied to different classes of learner such as Bayesian networks and K-means clustering.

Keywords: Inner Ensembles, Bayesian Network, K-means, Comprehensibility.

1 Introduction

The idea of the *wisdom of crowds* is that decisions made by groups of people are often more accurate, and more robust, than those made by individuals. An important sub-field in machine learning, ensemble methods, has exploited this idea very effectively, particularly in producing substantial performance gains. However, we argue that there is considerable room to extend it further. We believe our work is just the beginning of a much wider use of ensemble methods. Here, instead of combining the output of models, we apply ensemble methods to the decisions used to generate the models. We call this idea *Inner Ensembles* as the *wisdom of crowds* is applied inside the learning algorithm. Although this idea has been applied to decision trees [1,2], it is in fact much more general and has broader benefits than previously thought. Here we argue that like more traditional ensemble methods, *Inner Ensembles* define a broad framework that has the potential to impact all kinds of algorithms other than just decision trees.

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part III, LNAI 8190, pp. 33–48, 2013.

Using this framework, we can realize many of the advantages of traditional ensemble methods while restoring the more intuitive models produced by the base algorithms. Many of us have worked extensively with such models and we have a clear sense of what has been learned. This is particularly important when the task is knowledge discovery rather than prediction. On a more practical level, Inner Ensembles produce models with a number of clear advantages: comprehensibility, stability, simplicity, fast classification and small memory footprint. Certainly, many of these are problematical for traditional ensemble methods [3]. However, we recognize that these advantages must often be traded off against absolute performance. The work reported here shows that much of the improved performance can be maintained. Continued refinement of the approach should lead to further improvement.

Comprehensibility, how understandable a model is to users, is essential in many real-world problems: medicine, fraud detection in insurance companies, loan concession in financial environments [2]. Comprehensibility acts as a validation tool in some domains such as medical diagnosis; users are confident in a system only when they understand how it arrives at decisions [4]. A comprehensible model helps in identifying important hidden feature relationships. It may suggest better representations, improving an algorithm's generalization power [5]. Finally, comprehensibility may help to refine "approximately-correct"' domain theories [6]. Comprehensibility is an important feature of inner-ensembles, but not the only one. Stability is the property of being robust to small changes in the underlying data [7]. Robust models are important because they evoke more confidence that the underlying concept has been truly captured and their accuracy is less susceptible to noise. Simplicity is an important property in its own right, typically motivated by Occam's razor [8]. The closely related concept of over-fitting avoidance has been an important issue within machine learning for some years. It is not without controversy though and the exact reasons for the desirability are open to question [9]. Simplicity in terms of the actual features used also leads to another two desirable properties: fast classification and small memory footprint. In many on-line applications, the speed of determining membership, either in classification or clustering, is an important practical consideration [10].

There has been quite a bit of work addressing the shortcomings of the standard ensemble method, particularly the lack of comprehensibility. There are generally two directions that have been followed. The first uses an ensemble as part of the predictive model gaining the comprehensibility through the simplified high-level structure [11,12]. The second uses a standard ensemble as a guide to growing a new and simpler model that is comprehensible [13,14,15]. However, our approach, Inner Ensembles, is quite different, using the ensemble to chose the parts of the model. To illustrate the point, standard ensemble learning is analogous to a management meeting in a company where decisions are made by voting; Inner Ensembles learning is analogous to one manager being selected by voting to make the decision on behalf of the rest of the group. To the best of our knowledge there are two pieces of work that can be considered as Inner Ensembles [1,2]. However,

these two were focused solely on decision trees and then just on choosing the right feature to split. What we are doing is generalizing this idea to work for many different kinds of algorithms. Certainly, our framework offers additional benefits. To support this claim we introduce general guidelines for using Inner Ensembles which we have applied to two categories of learning: supervised and unsupervised. For the former we use Bayesian networks, for the latter K-means clustering. We use ensemble methods similar to bagging. In the future work section, we discuss the other potential ways of extending this idea especially using boosting instead of bagging and using Inner Ensembles for other kinds of algorithms. In the rest of the paper, we begin by explaining how to apply the framework for Inner Ensembles to existing algorithms. Next, we present experiments that show the efficacy of our framework. Finally, we will draw conclusions and suggest how future work will explore new applications for Inner Ensembles.

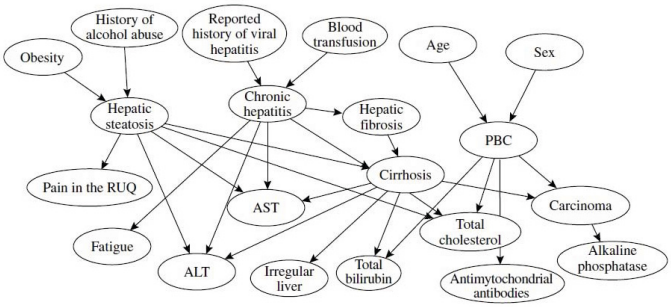

Fig. 1. An example of a liver disorder diagnosis network [16]

2 Inner Ensembles in Practice

In this section, we describe general guidelines for applying Inner Ensembles to any algorithms. Then using these guidelines, we describe in detail how our framework is applied to two quite different algorithms: Bayesian networks and K-means clustering. By choosing both a supervised and unsupervised algorithm, we aim to demonstrate the broad applicability of our framework. As the structure of each algorithm is different, it is difficult to define a precise method for applying Inner Ensembles in every case. However, with the knowledge of how each algorithm works, we can follow some general guidelines. We need to identify decision points, where choices are made based on a measure. Then, we need to locate the input and output of that decision maker. By manipulating the input, say by sampling the data, we produce different outputs. We combine these outputs and apply the result to the decision being made at that point inside the algorithm. This procedure is shown in algorithm 1.

Algorithm 1. General guidelines for using Inner Ensembles

1: **Locating a decision maker inside the algorithm.**
2: **Finding a measure based on which that decision maker works.**
3: **Indicating the input and output of the measure.**
4: **Applying the ensemble on the measure:**
5: Changing the input of the measure in different ways. {It can be sampling of data or feature based modifications or other methods.}
6: Generating an output based on each input.
7: Combining the outputs
8: **Applying the output of the ensemble for decision making.**

2.1 Learning Bayesian Network Structure

Bayesian networks have been used for many applications: medicine, expert systems [17] and path finder systems [18]. One of the major advantages is their comprehensibility [19]. This advantage would clearly be lost when multiple networks form an ensemble. Let us illustrate this point using the real world application of liver disorder diagnosis [16]. The network, shown in figure 1, consists of the risk factors and symptoms for several related disorders and it is important that it be understandable for a clinician. For example, the network shows that *Alcohol Abuse* and *Obesity* are risk factors for *Hepatic Steatosis*, a fatty liver. *Hepatic Steatosis*, itself, may produce *Pain* and is linked to *Cirhosis*. When using an ensemble of Bayesian networks, we would have many such networks with different numbers of arcs which may represent quite different relationships; it would be difficult for clinician to understand such a model. By using Inner Ensembles, we gain many of the benefits of the traditional ensemble method while keeping the comprehensibility of the base algorithm.

Bayesian networks specify a set of conditional independence assumptions; this is captured in the structure of the network. To completely specify the network, we also need conditional probability tables [20]. Therefore, for Bayesian networks, there are two distinct learning problems, we focus on the former. Specifically, we apply our framework to build the structure of the network during the learning phase of the algorithm. Bayesian network algorithms search for the best network structure among all possible ones. The popular K2 algorithm, shown in Algorithm 2, uses a scoring function to determine the better of two networks and a given ordering of nodes to determine the sequence in which they are processed [21]. For all the nodes in the ordered list (line 2), the algorithm begins with no parents for each node (line 3), π_i is the set of parents of the *ith* node. At each step (line 4), one parent is added to the node, and the score, $g(i, \pi_i \cup \{z\})$, of the network structure is calculated (line 5), $Prec(x_i)$ in algorithm 2 denote the nodes that precede node x_i in the ordered list. The parent that increases the score the most is added to the node's list of parents π_i. Adding parents to the node stops if the addition does not increase the score of the structure (lines 7-11). On completion, the algorithm produces a final structure.

We follow the general guidelines presented in algorithm 1 to apply Inner Ensembles to the Bayesian network algorithm. First we locate a decision maker.

Algorithm 2. K2 algorithm

1: **Input:** A set of n nodes, An ordering of the nodes, u maximum number of parents for each node, A database D containing N instances.
2: **for** i=1 to n **do**
3: $\pi_i = \phi$, $P_{old} = g(i, \pi_i)$, OKTOProceed = true.
4: **while** OKTOProceed **and** $|\pi_i| < u$ **do**
5: z is the node in $Pred(x_i)$ - π_i that maximizes $g(i, \pi_i \cup \{z\})$
6: $P_{new} = g(i, \pi_i \cup \{z\})$
7: **if** $P_{new} > P_{old}$ **then**
8: $P_{old} = P_{new}$, $\pi_i = \pi_i \cup \{z\}$
9: **else**
10: OKTOProceed = false
11: **end if**
12: **end while**
13: **write** (Node: x_i , Parents of this node: , π_i)
14: **end for**

Algorithm 3. LOO global score "g" for K2 algorithm.

1: $Acc = 0$.
2: $D = D_1, ...D_N$. {Instances}
3: **for** $i = 1$ to N **do**
4: Estimate conditional probability for network using existing structure and D-D_i.
5: $Acc = Acc + PredictAccuracy(D_i)$
6: **end for**
7: score is: $\frac{Acc}{N}$.

According to algorithm 2, it decides if a node can be added as a parent. The next step is to find a score function for this decision maker, $g(i, \pi_i \cup \{z\})$. There are two types: local and global [22]. We use the global score function g (line 3 algorithm 2) calculated as shown in algorithm 3. Here, using leave-one-out cross validation (LOO), the algorithm extracts one instance for validation, the rest of the data forming the training set (line 4). At each iteration, the network is built using the training set and tested on the single instance. The final score of the network is the average of the scores across all splits of the data (lines 5,7). The nest step according to algorithm 1 is to indicating the input and the output of the scoring function (LOO global score). The input is the training data and the output is the score. Then, line 5 of algorithm 1, we change the input of the measure by generating E sampling of the data for which E is the ensemble size. For each of those E samplings, the output of the LOO global score is calculated (line 6 algorithm 1) and finally the E outputs are combined by averaging (line 7). Using this procedure, the global score is redefined as shown in algorithm 4, we call this the $g_{Ensemble}$ score. Sampling of the data in algorithm 4 can be any kind of sampling. It can be with or without replacement. It can also be a different size with respect to the original dataset. Depending on the type of the sampling, we have different ensemble methods. For example, in the case of sampling with

Algorithm 4. Ensemble LOO global score "$g_{Ensemble}$" for K2 algorithm.

1: E {Ensemble Size}
2: $D = D_1, ...D_N$. {Instances}
3: $FinalScore = 0$.
4: **for** $k = 1$ to E **do**
5: $D_S = Sample(D, Size)$ $\{D_S = D_{S1}, ...D_{SSize}\}$
6: $FinalScore = FinalScore + LOOScore(D_S)$.
7: **end for**
8: score is: $\frac{FinalScore}{E}$

replacement and sample size of 100%, we have bagging. Once the best structure has been selected, the algorithm proceeds in the conventional fashion calculating the conditional probability tables necessary for the complete Bayesian network.

2.2 K-means Clustering

For unsupervised learning, we apply Inner Ensembles to K-means clustering, a very popular clustering method. One advantage of such a method is that it characterizes each cluster in terms of a single point, called a prototype. The prototype is a representative of all samples inside that cluster and thus the meaning of that cluster. The idea that a prototype is an important way of representing a particular concept is central to certain theories in Cognitive Science [23]. For our purposes, what is important about prototypes is that they help in the human understanding of the structure of a particular problem or domain. In medical diagnosis, a prototype might be a vector of numerical results from tests or measurements. A clinician assessing an individual's risk for a heart problems will combine measurements of blood pressure, cholesterol and weight. A prototype, identified in a clustering over many subjects, representing a high risk patient would be clearly separated from one of low risk. Prototypes offer other advantages such as use in compression and for efficient finding of the nearest neighbor [24]. Using a traditional ensemble of K-means clustering, we loose the prototypes which are the cluster centers for K-means. Using Inner Ensembles we keep the cluster prototype, and thus comprehesibility, while gaining the performance advantage of the ensemble method.

Algorithm 5 details the steps in K-means clustering [25]. It starts by initializing the cluster centers (line 1). Then, over several iterations, it assigns instances to the closest cluster center and updates the centers (lines 4-7). Finally, the algorithm stops when there are no more changes in the cluster centers (line 9). Following the lines of general guidelines, algorithm 1, we need to find a decision maker inside K-means algorithm that assigns each instance to the closest cluster center (line 1). Next we find a score function, the Euclidean distance in this case (line 2). Next we find the input and output of the score function (line 3). The input of the Euclidean distance is the cluster centers and the data; the output is the distance of the data to each cluster center. Thus we can change the input of the measure that is the data by generating E different sampling

Algorithm 5. K-means clustering algorithm.

1: Initialize K cluster centers $\mu_1, \mu_2, ..., \mu_K$
2: Data: $X = \{X_1, X_2, ..., X_N\}$
3: **repeat**
4: **for** All data instances X_i **do**
5: $m = argmin_K\{d(X_i, \mu_K)\}$ {Closest cluster center to each instance}
6: Assign instance X_i to cluster center μ_m
7: **end for**
8: Update cluster centers.
9: **until** (No assignment change or max iterations)
10: **Return** The clusters.

Algorithm 6. Inner K-means clustering algorithm

1: Initialize K cluster centers $\mu_1, \mu_2, ..., \mu_K$
2: Data: $X = \{X_1, X_2, ..., X_N\}$
3: **repeat**
4: **for** All data instances X_i **do**
5: $m = \{m_1, ..., m_N\}$.
6: **for** j= 1 to Ensemble size **do**
7: $S = Sample(FeatureSpace)$
8: $I = argmin_E\{d(X_{Sj}, \mu_E)\}$ {Closest cluster center to each instance}
9: $m_I = m_I + 1$
10: **end for**
11: $L = argmax_e(m_e)$ {Voting}
12: Assign instance X_i to cluster center μ_L
13: **end for**
14: Update cluster centers.
15: **until** (No assignment change or max iterations)
16: **Return** The clusters.

of the data (line 5). Although we initially used sampling of the instances, our experimental results were poor because it has little impact on the location of the centers. Thus we lose diversity that is very important for ensemble methods. To improve diversity, we use random subsets of features for each ensemble member. The algorithm repeatedly selects a random subset of features. Next the outputs of each generated input are calculated (line 6). In this case based on the feature subset, the closest cluster center is found using Euclidean distance. Therefore for each data instance, we have a set of candidate cluster centers according to different feature subsets. In the combining step (line 7), the cluster center with the most number of votes is selected for that particular data instance. This is shown in algorithm 6.

3 Experimental Results

In this section, we run several experiments for our new versions of the K2 Bayesian network and K-means clustering algorithms. We believe that Inner

Ensembles attain many of the benefits of traditional ensemble through similar means, i.e.. reducing the variance in the bias-variance trade-off [26]. So, we expect that whenever the ensemble methods work, Inner Ensembles will. Thus, the experiments test two different hypotheses: that our new versions are superior in performance to the base algorithms and that whenever the traditional ensemble method improves the performance, Inner Ensembles improves it too. For Bayesian networks, we compare the results with bagging because the Inner Ensembles sampling method we used is similar to bagging. For Inner K-means, we compare the results with several cluster ensemble methods in terms of different cluster validation measures. For both experiments, we use UCI repository datasets [27]. As the statistical test, we used Friedmans test and for post-hoc we use Nemenys test both with $\alpha = 0.05$ in all of the experiments. We report the results of a sensivity analysis study in the course of which different parameters of the algorithm are considered. These parameters include:

- *Ensemble Size:* 10 different ensemble sizes 10-100.
- *Sample Size:*10 different sampling sizes 10%-100%
- *Resampling Mode: With* or *without* replacement.

3.1 Inner Ensemble K2

In addition to the experiments noted above, for Bayesian networks we run another to confirm that in terms of classification time as well as in terms of comprehensibility our method is superior to bagging. We use 14 datasets from the UCI repository with different number of classes, features and instances. In all of the tables, BN is the original network, IEBN the Inner Ensemble Bayesian Network and BGBN the bagging of Bayesian networks.

For each of the parameters, we used 10 fold cross validation 10 times. For each run, the average error of the network is calculated. Table 1 shows the comparison of the accuracy for different ranges of parameters. In this table IEBN No Rep means sampling without replacement. Also S-Size and En-Size list the range of parameters for the sample size and the ensemble size for which IEBN performs better than BN. For better understanding, we include the percentage of parameters that results in better performance and we list the average and the best results for those parameters. For simplicity of presentation, we only report the results obtained for sampling without replacement because in this sampling regimen, IEBN works better than BN on 62% of the parameters while for sampling with replacement it only works on 46% of the parameters.

We begin by comparing the average performances of IEBN and BGBN with original BN. The null hypothesis for Friedman's test is BN , IEBN and BGBN perform similarly. The Friedman statistic is 26.14 for IEBN and BGBN while the critical value for Friedman's test is 6.00 for the 3 models. Thus we can reject the null hypothesis, there is at least one classifier with significant difference in performance. We use Nemenyi's test to rank the classifiers according to their performance. The critical value for Nemenyi's test is equal to 2.34 for the 3 models and the q-values for BN-IEBN and IEBN-BGBN are 2.83 and 2.27 respectively

Table 1. The Accuracy of the Various Algorithms

		IEBN No Rep			IEBN No Rep		BGBN	
Dataset	**BN**	**S-Size**	**En Size**	**Perc**	**Avrg**	**Best**	**Avrg**	**Best**
Breast-w	97.1	[10-90]	[10-100]	70%	97.2	97.31	97.18	97.18
Credit-a	85	[10-90]	[10-100]	99%	85.62	86.06	86.24	86.48
Ecoli	80.77	[10-90]	[10-100]	50%	81.03	81.7	85.35	85.92
Glass	70.84	[10-90]	[10-50]	4%	71.1	71.31	74.21	75.14
Heart-c	80.76	[10-90]	[10-100]	96%	81.63	82.74	82.56	82.94
Heart-S	82.19	[10-90]	[10-100]	32%	82.48	82.96	82.71	83.19
Hepatitis	82.39	[10-60]	[40-100]	10%	82.75	83.55	82.88	83.03
Iris	92.87	[10-90]	[10-100]	81%	93.24	93.8	94.58	94.93
Labor	88.25	[10-90]	[10-100]	99%	90.64	92.81	93.04	94.04
Liver	56.93	[10-90]	[10-100]	90%	57.41	58.41	63.75	64.58
Lymph	81.96	[10-90]	[10-100]	97%	83.19	85.14	84.66	85.41
Pima	74.66	[10-90]	[10-100]	97%	75.35	76.05	75.77	75.98
Tic-tac	92.44	[50-90]	[10-100]	18%	92.58	92.73	96.87	97.18
Vote	96.05	[30-90]	[10-100]	26%	96.12	96.25	96.15	96.21
Average	83			62%	83.59	84.34	85.42	85.87

which shows that the performance of BN on all the datasets is significantly different from that of IEBN. On the other hand the performance of IEBN and BGBN is not significantly different. We also use Friedman's test to compare the BN with the best results of IEBN and the best results of BGBN. Here the Friedman statistic is 22.29 and the critical value for Friedman's test is 6.00 for the 3 models. Thus we can reject the null hypothesis. By performing Nemenyi's test, the q-value for BN-IEBN and IEBN-BGBN are 3.4 and 1.13 which confirm that IEBN is significantly different from BN but not from BGBN. Generally we can see that with sampling without replacement, on 62% of the entire parameters IEBN works significantly better than BN on all of the datasets.

Generally, as expected, IEBN improves the performance over the original network whenever Bagging does but the improvement is smaller. Therefore we are not expecting improvements on Breast-w and Vote dataset because bagging does not. For individual datasets on which bagging improves the performance , except for Tic-tac-toe, the performance improves over that of the base algorithm. However the gain for Ecoli and Glass datasets are small. Future work on the way of creating ensembles inside the Bayesian network, will determine how closely our method can approach bagging in terms of performance. At present, our method is just applied to learning the structure of the network. One possible avenue for future work is to apply Inner Ensembles for the second part of Bayesian network learning, estimating conditional probabilities.To compare the classification times of Inner Ensembles and bagging, we ran the experiment on all of the datasets for different ensemble sizes, and averaged over all of the datasets. Figure 2 shows the average time for different ensemble size for BGBN versus IEBN. As we can see from this figure, the classification time is a lot faster for Inner Ensembles than bagging. An increase in the ensemble size in the case of Inner Ensembles

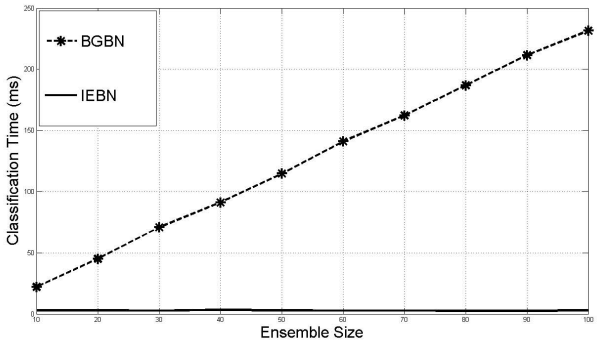

Fig. 2. Classification time (mS) for BGBN vs IEBN

has little effect on classification time because the number of models remains the same.

So in this section, we have shown how our method achieves some of the performance improvement of bagging while maintaining the comprehensibility, classification performance and fast classification time of the original Bayesian network.

3.2 Inner Ensemble K-means

To test our new version of K-means, we compare results on 15 UCI datasets, each with the necessary numerical attributes used by such a clustering system. Among possible different performance measures for clustering, we use Normalized Mutual Information that determines the shared information between the clusterings [28] and Purity that measures the coherence of a cluster [29]. Equation 1 shows NMI for which $I(X, Y)$ is the mutual information between two random variables X, Y; $H(X)$ is the entropy of X; X is the result of the clustering; Y contains the true labels. Equation 2 shows the cluster Purity. In this equation, n is the total number of instances, m the number of clusters and P_j is the fraction of the cluster to which the largest class of objects is assigned. For both measures, the larger the value the better the results.

$$NMI(X, Y) = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}} \tag{1}$$

$$Purity = \sum_{j=1}^{m} \frac{n_j}{n} P_j \tag{2}$$

For these experiments, we compare the performance of inner K-means with two different types of cluster ensemble method. The first is a graph-based approach (G-Based) that includes three different methods CSPA, MCLA and HGPA [28]. The second is based on pair-wise similarity. The methods of this latter group first build a similarity matrix, called a co-association matrix and then uses it for different forms of hierarchical clustering. This group is also called

Hierarchical Agglomerative Clustering Consensus (HAC) [30]. There are three different methods for HAC based on the hierarchical clustering: single, average and complete linkage.

For simplicity, we do not report the results of all 6 ensemble methods. Instead, for the graph based methods, we report the best performance of CSPA, MCLA and HGPA. For the HAC methods, we report the best performance among the different linkage types. For each of the parameters (ensemble size, sample size), we run Inner Ensemble, cluster ensemble and original K-means 30 times with 30 different initial cluster centers and the average performance is calculated. For a fair comparison, we run original K-means with the same cluster centers as used for Inner Ensembles. As the Inner Ensembles uses different subsets of features, we generate the members for cluster ensemble the similar way. If the number of instances are larger than 500, we sample the data without replacement until 500 is reached.

In table 2, we report the results of a sensitivity analysis using a range of parameters. We only report the results obtained with sampling with replacement, using this resampling strategy, Inner Ensemble K-means (IEK-m) works better than K-means on 55% of the parameters while for sampling without replacement it just works on 53% of the parameters. From this table, we can see that IEK-m generally outperforms K-means, the graph-based and HAC methods in terms of the NMI measure. For the Heart-s, Pima and Sonar datasets, the NMI is close to zero, the clusters that where extracted from these datasets are nothing like the true classes. For Sonar and Vehicles and Yeast the cluster ensemble methods do not improve the performance and, as we expected, IEK-m does not improve it either. IEK-m improves the performance on the rest of the datasets except for Ecoli for which one of the ensemble methods decreases the performance. In comparison, HAC improves the performance on just 8 out of 15 datasets, does not improve it on 7 .G-based methods improve the performance on 9 out of 15, do not improve it on 3 and decrease it on 3 datasets.

To compare the performances all datasets, first we compare the average performance of K-means, IEK-m, G-Based and HAC. The Friedman statistics for the average case for K-means, IEK-m, G-Based and HAC is 9.66, larger than the critical value 7.5 for 4 models. The q-value for K-means and IEK-m for NMI is 2.75, larger than the critical value which is 2.57 for 4 models. This shows that IEK-m is significantly different from K-means on all datasets in the average case. Also for the NMI measure, the q-values for HAC and G-Based are 0.14 and 1.06 respectively which are both less than 2.57 showing that that for the average case, IEK-m is not significantly different from G-Based and HAC. To compare the best results, the Friedman statistics for the best case is 16.5 which is larger than the critical value 7.5 for 4 models. The q-value for K-means and IEK-m for NMI is 3.88 which is larger than the critical value which is 2.57 for 4 models. This shows that for the best case IEK-m is significantly different from the K-means on the entire data. Also for the NMI measure in the best case, the q-values for HAC and G-Based are 1.06 and 2.12 which both are less than 2.57 . In the best case , IEK-m is not significantly different from G-Based and HAC.

Table 2. Comparison of K-means, Inner Ensemble K-means and Ensemble K-means (G-Based, HAC) in terms of NMI

		IEK-m Rep			IEK-m Rep		G-Based		HAC	
Dataset	**K-means**	**S-Size**	**En Size**	**Perc**	**Avrg**	**Best**	**Avrg**	**Best**	**Avrg**	**Best**
Ecoli	0.59	[-]	[-]	0%	0.59	0.59	0.57	0.57	0.61	0.62
Glass	0.38	[40-90]	[10-100]	49%	0.4	0.42	0.34	0.34	0.38	0.38
Heart-s	0.02	[20-90]	[10-100]	88%	0.11	0.31	0.04	0.05	0.02	0.02
Iris	0.71	[50-90]	[10-100]	48%	0.75	0.8	0.77	0.82	0.77	0.79
Letter	0.44	[30-90]	[10-100]	63%	0.45	0.46	0.47	0.47	0.47	0.48
Optdig	0.72	[20-90]	[10-100]	86%	0.74	0.75	0.75	0.76	0.75	0.77
Pendig	0.67	[40-90]	[10-100]	54%	0.68	0.7	0.68	0.69	0.69	0.7
Pima	0.02	[30-90]	[10-100]	68%	0.05	0.09	0.11	0.11	0.02	0.02
Segment	0.54	[20-90]	[10-100]	67%	0.59	0.63	0.56	0.58	0.61	0.62
Sonar	0.01	[10-90]	[10-100]	66%	0.01	0.02	0.01	0.01	0.02	0.02
Vehicle	0.19	[20-90]	[10-100]	80%	0.19	0.2	0.19	0.19	0.19	0.2
Vowel	0.2	[20-80]	[10-100]	34%	0.24	0.29	0.21	0.21	0.21	0.21
Wavef	0.36	[20-90]	[10-100]	53%	0.37	0.38	0.36	0.37	0.36	0.37
Wine	0.43	[20-90]	[10-100]	69%	0.55	0.72	0.45	0.51	0.43	0.43
Yeast	0.28	[90-90]	[70-70]	1%	0.28	0.28	0.27	0.27	0.28	0.28
Average	0.37			55%	0.4	0.44	0.39	0.4	0.39	0.39

Table 3. Comparison of K-means, Inner Ensemble K-means and Ensemble K-means (G-Based, HAC) in terms of purity

		IEK-m Rep			IEK-m Rep		G-Based		HAC	
Dataset	**K-means**	**S-Size**	**En Size**	**Perc**	**Avrg**	**Best**	**Avrg**	**Best**	**Avrg**	**Best**
Ecoli	0.81	[90-90]	[30-90]	6%	0.81	0.82	0.8	0.8	0.81	0.82
Glass	0.57	[30-90]	[10-100]	49%	0.58	0.6	0.6	0.62	0.57	0.57
Heart-s	0.59	[20-90]	[10-100]	88%	0.68	0.81	0.62	0.63	0.59	0.59
Iris	0.83	[50-90]	[10-100]	56%	0.86	0.9	0.91	0.94	0.9	0.92
Letter	0.32	[20-90]	[10-100]	76%	0.34	0.35	0.34	0.35	0.34	0.35
Optdig	0.74	[10-90]	[10-100]	96%	0.77	0.79	0.81	0.82	0.78	0.8
Pendig	0.69	[40-90]	[10-100]	62%	0.71	0.73	0.72	0.73	0.7	0.71
Pima	0.66	[30-80]	[10-100]	48%	0.67	0.7	0.68	0.69	0.66	0.66
Segment	0.57	[20-90]	[10-100]	79%	0.61	0.64	0.62	0.66	0.64	0.65
Sonar	0.55	[10-90]	[10-100]	66%	0.56	0.58	0.55	0.56	0.55	0.56
Vehicle	0.45	[20-90]	[10-100]	84%	0.46	0.47	0.47	0.47	0.47	0.47
Vowel	0.23	[20-80]	[10-100]	46%	0.26	0.31	0.24	0.24	0.23	0.23
Wavef	0.56	[10-90]	[10-100]	40%	0.56	0.59	0.55	0.55	0.56	0.56
Wine	0.7	[20-90]	[10-100]	71%	0.81	0.92	0.75	0.78	0.7	0.7
Yeast	0.52	[50-90]	[50-80]	8%	0.52	0.52	0.52	0.52	0.54	0.54
Average	0.59			58%	0.61	0.65	0.61	0.62	0.6	0.61

Table 3 shows the results of the clustering in terms of the purity measure. We report the results obtained for sampling with replacement because this sampling regimen, IEK-m works better than BN on 58% of the parameters while for sampling without replacement it only works on 53% of the parameters. IEK-m generally outperforms both K-means and ensemble methods. For the average case IEK-m improves the performance on most i.e. 12 out of 15 datasets and for the best case 14 out of 15 datasets. Also in the average case , the HAC methods improve the performance on 7 out of 15 datasets and on the best case 10 out of 15, but not on the rest. The average case for the G-based methods improves the performance on 11 out of 15, decrease the performance of Waveform and Ecoli and with no improvement on the rest and for the best case,G-Based improves the performance on 12 out of 15 dataset and it is otherwise to the average case.

The Friedman statistics for the average case is 11.5 which is larger than the critical value 7.5 for 4 models. On average, the q-value for K-means and IEK-m for Purity is 2.89 which is larger than the critical value which is 2.57 for 4 models. This shows that IEK-m performs significantly better than K-means on all datasets. For the Purity measure, in the average case, the q-values for HAC and G-Based are 0.84 and 0.07 respectively which are both larger than 2.57. Thus IEK-m is not significantly different from G-Based and HAC. The Friedman statistics for the best case is 17.00 larger than the critical value 7.5 for 4 models. In the best case, the q-value for K-means and IEK-m for Purity is 3.81 which is larger than the critical value of 2.57. Thus IEK-m performs significantly better than K-means on all datasets. Also for the best case, the q-values for HAC and G-Based are 1.55 and 0.56 both larger than 2.57. Thus IEK-m is not significantly different from G-Based and HAC.

To sum up, our experiments show that our framework is broadly applicable on different types of algorithms. More specifically, we apply our framework on two groups of supervised and un-supervised learning. Our experiments generally show that: (1) As we expected, wherever ensemble methods work, our framework improves the performance on both the supervised and the un-supervised cases. (2) Our framework improves the performance on a large portion of the parameters. (3) For both supervised and un-supervised learning, our framework improves the performance significantly over the original method. But it does not improve the performance over ensemble methods for supervised learning. However, for un-supervised learning, our results shows that Inner Ensembles works comparably or are superior to the regular ensembles.

4 Limitations and Future Work

One limitation of the work is that the reported results are based on the parameters for which Inner Ensembles improve the performance. A better way of doing the experiments is to find the best parameters prior to evaluating performance.To address this issue, we have run some initial experiments using cross-validation to find the best parameters before running the algorithm on the test data. So far we have only done this for K-means where we obtained very similar results

to those presented earlier. To validate the stability, we run IEK-m on different noisy data. We add random noise to the most 50% significant features of the datasets from 20%, 40%, 60% and the primary results shows that the IEK-m is more stable than original K-means and generally comparable to cluster ensembles on 20% and 40% noise. On 60% some of the cluster ensemble methods are more stable than IEK-m but still IEK-m is more stable than original K-means.

So far, we do the experiments to confirm the performance of the Inner Ensembles. But as we mentioned in the introduction there are other advantages of using Inner Ensembles such as stability, classification time and small memory usage. We need further experiments to confirm the stability of the IEBN and a discussion about the memory usage of the Inner Ensembles and speed of IEK-m which we thing can be faster in online clustering. For future work, we want to explore other classification and clustering algorithms. Nor are we restricted solely to those situations, we can apply this idea to a variety of other algorithms, such as the type of search used in learning or the pruning method for decision trees or other groups of methods such as feature selection. One important factor that affects the performance of the ensemble method is diversity. For future work the effect of diversity on the performance of Inner Ensembles needs to be investigated too. The idea can be extended in other ways. First we can improve the specific methods discussed in this paper by using different kind of samplings, such as weighted ones. So far we just used voting, akin to bagging. However, we intend to investigate different kinds of ensemble methods, such as boosting.

5 Conclusion

The main objective of this paper was to extend the possible ways that the machine learning community makes use of ensemble methods. Our particular approach is called Inner Ensembles. By using this new approach on supervised and unsupervised learning algorithms, we showed that this idea is broadly applicable. For supervised learning, we applied our method to the learning of the structure of Bayesian networks, a popular classification algorithm; for unsupervised learning we used it for K-means clustering, a popular clustering method. In the case of Bayesian network, Inner Ensembles work generally better than original Bayesian network but worse than bagging. On the other hand, for K-means, inner K-means performs better than original K-means and comparable to two families of clustering ensemble. We introduced this idea as a framework that has the potential of of being applicable in many different ways other than those we have discussed in this paper.

References

1. Geurts, P., Wehenkel, L.: Investigation and Reduction of Discretization Variance in Decision Tree Induction. In: Lopez de Mantaras, R., Plaza, E. (eds.) ECML 2000. LNCS (LNAI), vol. 1810, pp. 162–170. Springer, Heidelberg (2000)

2. Prez, J.M., Muguerza, J., Arbelaitz, O., Gurrutxaga, I.: A New Algorithm to Build Consolidated Trees: Study of the Error Rate and Steadiness. In: Proceedings of the International Intelligent Information Processing and Web Mining Conference. Advances in Soft Computing, pp. 79–88 (2004)
3. Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: From Ensemble Methods to Comprehensible Models. In: Proceedings of the 5th International Conference on Discovery Science, pp. 165–177 (2002)
4. Wolberg, W.H., Street, W.N., Mangasarian, O.L.: Machine Learning Techniques to Diagnose Breast Cancer from Image-Processed Nuclear Features of Fine Needle Aspirates. Cancer Letters 77, 163–171 (1994)
5. Flann, N.S., Dietterich, T.G.: Selecting Appropriate Representations for Learning from Examples. In: Proceedings of the 5th National Conference on Artificial Intelligence, pp. 460–466 (1986)
6. Craven, M.W., Shavlik, J.W.: Extracting Comprehensible Concept Representations from Trained Neural Networks. In: Working Notes on the International Joint Conference on AI Workshop on Comprehensibility in Machine Learning, pp. 61–75 (1995)
7. Dwyer, K., Holte, R.: Decision Tree Instability and Active Learning. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 128–139. Springer, Heidelberg (2007)
8. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Occam's Razor. Information Processing Letters 24(6), 377–380 (1987)
9. Domingos, P.: Occam's Two Razors: The Sharp and the Blunt. In: Knowledge Discovery and Data Mining, pp. 37–43 (1998)
10. Williams, N., Zander, S., Armitage, G.: A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. Special Interest Group on Data Communication 36(5), 5–16 (2006)
11. Zimmermann, A.: Ensemble-Trees: Leveraging Ensemble Power Inside Decision Trees. In: Proceedings of the 11th International Conference on Discovery Science, pp. 76–87 (2008)
12. Lou, Y., Caruana, R., Gehrke, J.: Intelligible Models for Classification and Regression. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 150–158 (2012)
13. Van Assche, A., Blockeel, H.: Seeing the Forest through the Trees: Learning a Comprehensible Model from a First Order Ensemble. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 418–429. Springer, Heidelberg (2007)
14. Zhou, Z.H., Jiang, Y., Chen, S.F.: Extracting Symbolic Rules from Trained Neural Network Ensembles. AI Communications 16(1), 3–15 (2003)
15. Domingos, P.: Knowledge Discovery Via Multiple Models. Intelligent Data Analysis 2, 187–202 (1998)
16. Pourret, O., Marcot, B., Naim, P.: Bayesian Networks: A Practical Guide to Applications. Statistics in Practice. Wiley, Chichester (2008)
17. Desmedt, J.: Computer-Aided Electromyography and Expert Systems. Clinical Neurophysiology Updates. Elsevier, Amsterdam (1989)
18. Heckerman, D.E., Nathwani, B.N.: Towards Normative Expert Systems: Part ii, Probability-Based Representations for Efficient Knowledge Acquisition and Inference Methods of Information in Medicine. In: Methods of Information in Medicine, pp. 106–116 (1992)
19. Vomlel, J.: Two Applications of Bayesian Networks. In: Proceedings of Conference Znalosti, pp. 73–82 (2002)

20. Mitchell, T.M.: Machine Learning, 1st edn. McGraw-Hill, Inc., New York (1997)
21. Cooper, G.F., Herskovits, E.: A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning 9(4), 309–347 (1992)
22. Bouckaert, R.: Bayesian Network Classifiers in Weka. Working paper series. Department of Computer Science, University of Waikato (2004)
23. Lakoff, G.: Women, Fire and Dangerous Things. University of Chicago Press, Chicago (1987)
24. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2005)
25. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. John Willey & Sons, New York (1973)
26. Breiman, L.: Bias, Variance, and Arcing Classifiers. Technical report, Statistics Department, University of California at Berkeley (1996)
27. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository (2007)
28. Strehl, A., Ghosh, J., Cardie, C.: Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions. Journal of Machine Learning Research 3, 583–617 (2002)
29. Rendón, E., Abundez, I.M., Gutierrez, C., Zagal, S.D., Arizmendi, A., Quiroz, E.M., Arzate, H.E.: A Comparison of Internal and External Cluster Validation Indexes. In: Proceedings of the American Conference on Applied Mathematics and the 5th International Conference on Computer Engineering and Applications, pp. 158–163 (2011)
30. Nguyen, N., Caruana, R.: Consensus Clusterings. In: Proceedings of the 7th International Conference on Data Mining, pp. 607–612 (2007)