Clusterer Ensemble

Zhi-Hua Zhou*, Wei Tang

National Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China

Abstract

Ensemble methods that train multiple learners and then combine their predictions have been shown to be very effective in supervised learning. This paper explores ensemble methods for unsupervised learning. Here an ensemble comprises multiple clusterers, each of which is trained by *k*-means algorithm with different initial points. The clusters discovered by different clusterers are aligned, i.e. similar clusters are assigned with the same label, by counting their overlapped data items. Then, four methods are developed to combine the aligned clusterers. Experiments show that clustering performance could be significantly improved by ensemble methods, where utilizing mutual information to select a subset of clusterers for weighted voting is a nice choice. Since the proposed methods work by analyzing the clustering results instead of the internal mechanisms of the component clusterers, they are applicable to diverse kinds of clustering algorithms.

Keywords: Machine Learning; Ensemble learning; Clustering; Unsupervised learning; Selective ensemble

1. Introduction

Clustering is a fundamental technique of unsupervised learning, where the task is to find the inherent structure from unlabeled data. A good clusterer should divide the data into several clusters so that the intra-cluster similarity is maximized while the inter-cluster similarity is minimized. Since such a technique is required everywhere and diverse inductive principles exist (Estivill-Castro, 2002), clustering is always an active area in machine learning.

During the past decade, ensemble methods that train multiple learners and then combine their predictions to predict new examples have been a hot topic (Dietterich, 2002). Since the generalization ability of an ensemble could be better than that of its component learners (Hansen & Salamon, 1990), it is not a surprise that ensemble methods have been widely applied to diverse domains such as face recognition (Huang *et al.*, 2000), optical character recognition (Drucker *et al.*, 1993), scientific image analysis (Cherkauer, 1996), medical diagnosis (Zhou *et al.*, 2002), *etc.*

It is worth noting that almost all ensemble methods are designed for supervised learning where the desired outputs, or labels, of the training instances are known. The known training labels are used in some ensemble methods such as AdaBoost (Freund & Schapire, 1995), to evaluate the component learners and then use the evaluation results to weight the learners and change the training data distribution. More importantly, the training labels are necessary for eliminating the ambiguity in combining the component predictions. For example, in voted classifiers, the votes for different class labels are counted and compared. Here it is trivial to determine which vote is for which class because the training labels have implicitly coordinated the component classifiers in the way that the *i*-th class labels of all the component classifiers are the same.

* Corresponding author. Tel.: +86-25-8368-6268; fax: +86-25-8368-6268. *E-mail address*: zhouzh@nju.edu.cn (Z.-H. Zhou).
Submitted: Apr.10, 2003; Accepted: Nov.12, 2005

The lack of training labels makes the design of ensemble methods for unsupervised learning much more difficult than that for supervised learning. For illustration, suppose there are two clusterers each has discovered three clusters from a data set, and the goal is to combine the clusterers so that data items are put into a same cluster if and only if they were put into a same cluster by both of the clusterers. This task is not trivial because there is no guarantee that the *i*-th cluster discovered by one clusterer corresponds to the *i*-th cluster discovered by the other clusterer. So, although ensemble has been well investigated in supervised learning, few works address the issue of designing ensemble methods for clustering.

In this paper, a process for aligning the clusters discovered by different clusterers is developed, which works by measuring the similarity between the clusters through counting their overlapped data items. Then, four methods for combining the aligned clusterers are proposed. They are *voting*, *weighted-voting* where the mutual information weights are used in voting, *selective voting* where the mutual information weights are used to select a subset of clusterers to vote, and *selective weighted-voting* where the mutual information weights are used not only in selecting but also in voting. Experimental results show that *selective weighted-voting* is the best method, whose performance is significantly better than that of a single clusterer. The experiments also reveal that profit is obtained by employing mutual information weights in voting, while greater profit is obtained by building selective ensembles.

The rest of this paper is organized as follows. Section 2 focuses on the generation of the component clusterers. Section 3 presents the align process and proposes methods for combining the aligned component clusterers. Section 4 reports on the experimental results. Finally, Section 5 summarizes the main contributions of this paper and raises several issues for future works.

2. Generate component clusterers

2.1. Notations

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbf{R}^d$ denotes an unlabeled data set in a feature space of dimension d. The i-th data item $\mathbf{x}_i$ is a d-dimensional feature vector $[x_{i1}, x_{i2}, \ldots, x_{id}]^{\mathbf{T}}$, where $\mathbf{T}$ denotes vector transpose. In order to simplify the discussion, here we assume that all the features are numerical, i.e. x_{ij} ($i = 1, \ldots, n, j = 1, \ldots, d$) is numerical.

A clusterer dividing $\mathbf{X}$ into k clusters could be regarded as a label vector $\lambda \in \mathbf{N}^n$, which assigns the data item $\mathbf{x}_i$ to the λ_i -th cluster, i.e. C_{λ_i} where $\lambda_i \in \{1, 2, \ldots, k\}$.

A clusterer ensemble with size t comprises t clusterers, i.e. $\{\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(t)}\}$, which could also be regarded as a label vector λ, $\lambda \in \mathbf{N}^n$ and $\lambda = \mathbf{F}(\{\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(t)}\})$ where $\mathbf{F}(\cdot)$ is a function corresponding to the combining methods presented in Section 3.

2.2. k-means

The idea of the well-known *k*-means algorithm (MacQueen, 1967) is to iteratively update the mean value of the data items in a cluster, and regard the stabilized value as the representative of the cluster. The basic algorithm is shown in Figure 1.

There exist a lot of variants to the basic *k*-means algorithm based on different distance measure or representation of the centers. Strehl *et al.* (2000) has shown that different distance measures have different impacts on the performance of *k*-means algorithm. For convenience of discussion, in this paper the basic *k*-means algorithm employing the Euclidean distance is used.

A characteristic of *k*-means algorithm is that it is quite sensitive to the choice of the initial points, i.e. the data items selected to be the initial centers of the clusters. In supervised learning, if an algorithm has several alternative parameter configurations, a simple strategy is to run the algorithm several times each with a specific configuration and then use a validation set to choose the best version. But in unsupervised learning, it is difficult to judge which version is the best since there are no training labels available. Fortunately, such a characteristic is

1. randomly select k data items as the centers of the clusters;
2. for each data item, assign it to the cluster whose center is the nearest one to the data item;
3. calculate the new center;
4. if there is no change, end the loop. Otherwise go to step 2.

Figure 1. The basic k-means algorithm

not a bad news for building ensembles of k-means, because now it is easy to obtain diverse component clusterers through simply running the algorithm multiple times with different initial points.

3. Combine component clusterers

3.1. Align process

The component clusterers must be aligned before they are combined. This is because the component clusterers may assign similar cluster with different labels. For example, suppose there are two clusterers, whose corresponding label vectors are $[1, 2, 2, 1, 1, 3, 3]^{\mathbf{T}}$ and $[2, 3, 3, 2, 2, 1, 1]^{\mathbf{T}}$, respectively. Although the appearance of these label vectors are quite different, in fact they represent the same clustering result. Therefore, the label vectors must be aligned so that the same label denotes similar cluster.

In this paper, the clusterers are aligned based on the recognition that similar clusters should contain similar data items. In detail, suppose there are two clusterers whose corresponding label vectors are $\lambda^{(a)}$ and $\lambda^{(b)}$, respectively, and each clusterer divide the data set into k clusters, i.e. $\{C_1^{(a)}, C_2^{(a)}, \ldots, C_k^{(a)}\}$ and $\{C_1^{(b)}, C_2^{(b)}, \ldots, C_k^{(b)}\}$, respectively. For each pair of clusters from different clusterers, such as $C_i^{(a)}$ and $C_j^{(b)}$, the number of overlapped data items, i.e. data items appear in both $C_i^{(a)}$ and $C_j^{(b)}$, is counted. Then, the pair of clusters whose number of overlapped data items is the largest, are matched in the way that they are denoted by the same label. Such a process is repeated until all the clusters are matched. The pseudo-code of the align process is shown in Figure 2.

```
For i = 1 to k, j = 1 to k Do
   OVERLAP_ij = Count(C_i^(a), C_j^(b))     /* OVERLAP is a k×k matrix, Count(A, B) is a function
                                               returning the number of overlapped data items in A and B */
Γ = Ø
While Γ ≠ {C_1^(b), C_2^(b), …, C_k^(b)} Do
   (u, v) = arg(max(OVERLAP_ij))     /* OVERLAP_uv is the maximal cell in the matrix */
   Match(C_u^(a), C_v^(b))           /* cluster C_v^(b) is matched to cluster C_u^(a)
   Delete OVERLAP_u*                 /* cells related to C_u^(a) are removed */
   Delete OVERLAP_*v                 /* cells related to C_v^(b) are removed */
   Γ = Γ ∪ {C_v^(b)}
End of While
```

Figure 2. The pseudo-code of the align process

When there are t ($t > 2$) clusterers, one clusterer could be regarded as the baseline to which the remaining clusterers are aligned. In this paper, the baseline clusterer is randomly selected from the component clusterers. Note that the align process requires only one-pass scan of the data items nevertheless how big the value of m is, and it requires the storage of only $(t\text{-}1)\times k^2$ integers that are used to keep the number of overlapped data items. It

is evident that such an align process is quite efficient.

It is worth noting that according to the objective optimized by some clustering algorithms such as *k*-means, different clusterers are similar if they have a similar clustering quality, i.e. if the sum of distances from data items to their nearest centers is about the same. However, since the goal of the process presented in this section is to enable the clusters generated by different clusterers be combined, nevertheless how similar the clusterers themselves are, it is the similar clusters instead of similar clusterers are to be identified.

3.2. Combining methods

The simplest combining method is *voting*, where the *i*-th component of the label vector corresponding to the ensemble, i.e. λ_i, is determined by the plurality voting result of $\lambda_i^{(1)}$, $\lambda_i^{(2)}$, …, $\lambda_i^{(t)}$.

The second method, i.e. *weighted-voting*, employs *mutual information* between a pair of clusterers (Strehl *et al.*, 2000) to compute the weight for each clusterer. For two label vectors, i.e. $\lambda^{(a)}$ and $\lambda^{(b)}$, suppose there are n objects where n_i are in cluster $C_i^{(a)}$, n_j are in cluster $C_j^{(b)}$, and n^{ij} are in both $C_i^{(a)}$ and $C_j^{(b)}$. The [0, 1]-normalized mutual information Φ^{NMI} can be defined as:

$$\Phi^{\mathrm{NMI}}\left(\lambda^{(a)},\lambda^{(b)}\right)=\frac{2}{n}\sum_{i=1}^{k}\sum_{j=1}^{k}n^{ij}\log_{k^2}\left(\frac{n^{ij}n}{n_i n_j}\right) \tag{1}$$

Then, for every clusterer, the average mutual information can be computed as:

$$\beta_m=\frac{1}{t-1}\sum_{l=1,l\neq m}^{t}\Phi^{\mathrm{NMI}}\left(\lambda^{(m)},\lambda^{(l)}\right) \qquad (m = 1, 2, \ldots, t) \tag{2}$$

The bigger the value of β_m is, the less statistical information contained by the *m*-th clusterer has not been contained by other clusterers. Therefore, the weights of the clusterers can be defined as:

$$w_m=\frac{1}{\beta_m \mathrm{Z}} \qquad (m = 1, 2, \ldots, t) \tag{3}$$

where **Z** is used to normalize the weights so that

$$w_m>0\left(m=1,2,\ldots,t\right) \quad and \quad \sum_{m=1}^{t}w_m=1 \tag{4}$$

It was shown that selective ensemble methods that select a subset of learners to ensemble may be superior to ensembling all the component learners (Zhou *et al.*, 2002a). The *mutual information weights*, i.e. $\{w_1, w_2, \ldots, w_t\}$, can be used to select the clusterers. This is realized by excluding from the ensemble the clusterers whose mutual information weight is smaller than a threshold. In this paper the threshold is set to $1/t$.

The selected clusterers can be combined via *voting*, or *weighted-voting* based on re-normalized mutual information weights of the selected clusterers. Thus, another two combining methods, i.e. *selective voting* and *selective weighted-voting*, are obtained.

It is worth mentioning that the time cost of *weighted-voting*, *selective voting*, and *selective weighted-voting* are comparable, while that of *voting* is slightly less because it does not require the computation of the mutual information weights. However, the time cost of computing the mutual information weights is negligible if comparing with that of the *k*-means clustering process. Therefore, the time cost of building an ensemble of *k*-means by the proposed methods is roughly *m* times of that of training a single *k*-means clusterer, where *m* is the number of clusterers that are trained to be considered for ensembling.

4. Experiments

4.1. Data sets

Ten data sets from the UCI Machine Learning Repository (Blake *et al.*, 1998) are used, all of which contains only numerical attributes except the class attributes. For *image segmentation*, a constant attribute has been removed. The information about the data sets is tabulated in Table 1. Note that the class attributes of the data sets have not been used in the training of the clusterers and the clusterer ensembles.

Table 1. Data sets used in experiments

data set	attribute	class	size
image segmentation	18	7	2,310
ionosphere	34	2	351
iris	4	3	150
liver disorder	6	2	345
page blocks	10	5	5,473
vehicle	18	4	846
waveform21	21	3	5,000
waveform40	40	3	5,000
wine	13	3	178
wpbc	33	2	198

4.2. Evaluation scheme

In general, it is difficult to evaluate a clusterer because whether its clustering quality is good or not almost fully depends on the view of the user. However, when a class attribute that has not been used in the training process exist, the scheme proposed by Modha and Spangler (2003) could provide a relatively objective evaluation, which assumes that the class attribute exposes some inherent property of the data set that should be captured by the clusterer.

In detail, the clusterers are converted into classifiers using the following simple rule: identify each cluster with the class that has the largest overlap with the cluster, and assign every data item in that cluster to the found class. The rule allows multiple clusters to be assigned to a single class, but never assigns a single cluster to multiple classes. Suppose there are c classes, i.e. $\{C_1, C_2, \ldots, C_c\}$, in the ground truth classification. For a given clusterer, by using the above rule, let a_h denote the number of data items that are correctly assigned to the class C_h. Then, the clustering performance of the clusterer can be measured by *micro-precision*, which can be computed as:

$$\text{micro-p} = \frac{1}{n}\sum_{h=1}^{c} a_t \tag{5}$$

The bigger the value of micro-p, the better the clustering performance.

Such a scheme can only be used to compare clusterers with a fixed number of clusters, i.e. clusterers with the same *model complexity*. Therefore, in our experiments, for a given data set, the value of k, i.e. the number of clusters to be discovered, is fixed to the number of classes conveyed by the class attribute. Note that the ensemble methods proposed in Section 3 do not guarantee that k won't be reduced after the combination process. In fact, in some cases such a reduction may be helpful because it may reveal that the number of actual clusters is smaller than that was anticipated. But here the reduction will disable the above scheme from comparing the clustering performance. Fortunately, such a reduction never occurs in all of our experiments.

4.3. Results

In our experiments the number of iteration steps of the *k*-means algorithm is set to 100, and the error improvement threshold is set to 1e-5. For each data set, each of the four ensemble methods proposed in Section 3 is used to build five clusterer ensembles comprising 5, 8, 13, 20, or 30 component clusterers, respectively. The process is repeated for 10 times. Then, for each data set, each method, and each ensemble size, the average micro-p and its standard deviation are recorded. The average performance of single *k*-means is also recorded for comparison. The detailed experimental results are presented in the Appendix of this paper.

The pairwise two-tailed *t*-test results under significance level of 0.05 are summarized in Table 2, where 'win/ loss' means that the ensemble method is significantly better/worse than the single *k*-means algorithm, and 'tie' means that there is no significant difference between the ensemble method and the single *k*-means algorithm.

Table 2. Summary of the pairwise two-tailed *t*-test results under significance level of 0.05. *w-voting* denotes *weighted-voting*, *sel-voting* denotes *selective voting*, and *sel-w-voting* denotes *selective weighted-voting*.

ensemble size	*voting*	*w-voting*	*sel- voting*	*sel-w-voting*
	win/tie/loss	win/tie/loss	win/tie/loss	win/tie/loss
5	1/7/2	2/6/2	3/4/3	4/4/2
I8	0/8/2	3/6/1	4/5/1	5/4/1
13	2/6/2	2/6/2	2/7/1	4/6/0
20	1/6/3	3/4/3	5/3/2	5/3/2
30	1/5/4	3/4/3	5/3/2	5/3/2

Table 2 shows that the clustering performance of *voting*, *weighted-voting*, and *selective voting* is worse than, comparable to, and slightly better than that of the single *k*-means clusterer, respectively, while the performance of *selective weighted-voting* is significantly better than that of the single *k*-means clusterer. It is impressive that when the ensemble size is 13, *selective weighted-voting* never loses to single *k*-means. This observation shows that ensemble methods could improve the clustering performance. It also reveals that utilizing mutual information in the combination of the component clusterers is beneficial. So does building selective ensembles.

Table 3 summarizes the best ensemble method for a given data set. It justifies that *selective weighted-voting* is the best method, which achieves the best performance on four data sets. Moreover, Table 3 shows that the performances of *weighted-voting* and *selective voting* are very close because each of them achieves the best result on three data sets.

Table 3. The best ensemble method for the experimental data sets. *w-voting* denotes *weighted-voting*, *sel-voting* denotes *selective voting*, and *sel-w-voting* denotes *selective weighted-voting*. Number in the bracket is the ensemble size with which the best clustering performance is obtained. For selective ensemble methods, the ensemble size is shown as a ratio of the number of selected clusterers against the number of clusterers available.

data set	best method	data set	best method
image segmentation	*sel-w-voting* (13/30)	*vehicle*	*sel-w-voting* (1.4/5)
ionosphere	*sel-w-voting* (4.5/30)	*waveform21*	*w-voting* (13)
iris	*w-voting* (20)	*waveform40*	*sel-voting* (3.5/8)
liver disorder	*sel-w-voting* (3.3/13)	*wine*	*w-voting* (8)
page blocks	*sel-voting* (4.1/13)	*wpbc*	*sel-voting* (2.0/30)

It is worth mentioning that although utilizing mutual information and building selective ensembles are comparably effective from the aspect of improving clustering performance, we believe that the latter mechanism provides bigger profit because it uses fewer component clusterers to make up an ensemble. In fact, Table 4

shows that the selective ensemble methods, i.e. *selective-voting* and *selective weighted-voting*, only keep about 28% to 40% available clusterers. In cases where the clusterers must be stored for future use, such an advantage of selective ensemble should not be neglected.

Table 4. The geometrical mean percentage of clusterers selected by *selective voting* and *selective weighted-voting* under different ensemble sizes.

ensemble size	percentage of selecting
5	39.2% (1.96/5)
8	38.1% (3.05/8)
13	33.7% (4.38/13)
20	31.8% (6.36/20)
30	28.0% (8.41/30)

The impacts of the change of ensemble size on the clustering performance are depicted in Figure 3. It is interesting to see that for methods that could improve the clustering performance, i.e. *weighted-voting*, *selective voting*, and *selective weighted-voting*, the performance increases as the ensemble size increases, but for the useless method *voting*, the performance keeps almost constant or even decreases as the ensemble size increases. Why the performance of *voting* is so poor is a problem to be explored in future works.

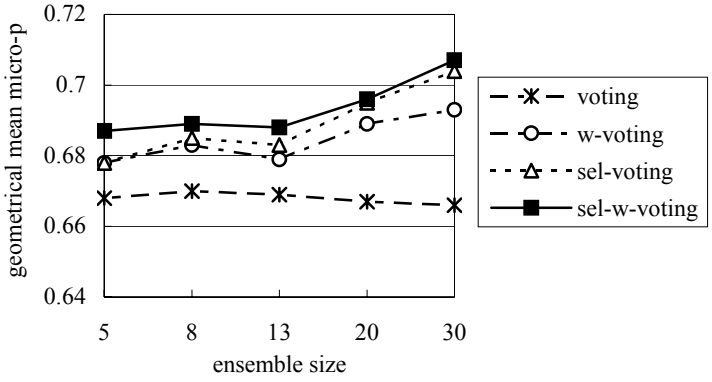

Figure 3. The impacts of the change of ensemble size on the clustering performance. *w-voting* denotes *weighted-voting*, *sel-voting* denotes *selective voting*, and *sel-w-voting* denotes *selective weighted-voting*. *geometrical mean* denotes the average across all the data sets. Here the *ensemble size* of *selective voting* and *selective weighted-voting* denotes the number of candidate clusterers instead of their real ensemble size.

5. Conclusion

In this paper, four methods are proposed for building ensembles of *k*-means clusterers. The component clusterers are generated by running the *k*-means algorithm multiple times with different initial points. An align process is applied to ensure that the same label used by different clusterers denotes similar clusters. The aligned clusterers are combined via *voting* or its variants. Experiments show that *selective weighted-voting* that utilizes mutual information to select a subset of clusterers for weighted voting is the best method, which could significantly improve the clustering performance. It is also found that utilizing mutual information weights or building selective ensembles are both beneficial to clusterer ensemble, while the latter mechanism is more rewardful because it could help obtain ensembles with smaller sizes.

It is worth mentioning that although *k*-means is used as the base clusterer in this paper, it does not mean that

the proposed methods can only be applied to *k*-means. Since these methods work by analyzing the clustering results instead of the internal mechanisms of the component clusterers, they are applicable to diverse kinds of clustering algorithms.

From the literatures on ensemble learning, it could be found that *voting* is an effective combining method that is often used in building ensembles of supervised learning algorithms. However, this paper shows that *voting* performs quite poor while its variants such as *selective weighted-voting* perform well in unsupervised learning scenario. How to explain this phenomenon remains an open problem to be explored in future works.

Another interesting issue to be explored is to see whether successful supervised ensemble methods, such as Bagging (Breiman, 1996) and AdaBoost (Freund & Schapire, 1995), can be modified for unsupervised learning. Moreover, since different distance measures have different impacts on the clustering performance (Strehl *et al.*, 2000), and different clustering algorithms may favor different kinds of cluster architectures, i.e. different algorithms may be effective at detecting different kinds of clusters, it will be interesting to investigate heterogeneous clusterer ensembles, i.e. ensembles composed of different kinds of clusterers.

Acknowledgement

This work was supported by the National Science Fund for Distinguished Young Scholars of China under the Grant No. 60325207, The Fok Ying Tung Education Foundation under the Grant No. 91067, and the Excellent Young Teachers Program of MOE, China.

References

Blake C., Keogh E., & Merz C. J. (1998). UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.htm], Department of Information and Computer Science, University of California, Irvine, CA.

Breiman L. (1996). Bagging predictors. *Machine Learning*, 24(2): 123-140.

Cherkauer K. J. (1996). Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks. *Proceedings of the 13th AAAI Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, Portland, OR, pp.15-21.

Dietterich T. G. (2002). Ensemble learning. In Arbib M. A. (Ed.), *The Handbook of Brain Theory and Neural Networks* (2nd edition). Cambridge, MA: MIT Press.

Drucker H., Schapire R., & Simard P. (1993). Improving performance in neural networks using a boosting algorithm. In Hanson S. J., Cowan J. D., & Lee Giles C. (Eds.), *Advances in Neural Information Processing Systems 5*, San Mateo, CA: Morgan Kaufmann, pp.42-49.

Estivill-Castro V. (2002). Why so many clustering algorithms – a position paper. *SIGKDD Explorations*, 4(1): 65-75.

Freund Y., & Schapire R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Proceedings of 2nd European Conference on Computational Learning Theory*, Barcelona, Spain, pp.23-37.

Hansen L. K., & Salamon P. (1990). Neural network ensembles. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(10): 993-1001.

Huang F. J., Zhou Z.-H., Zhang H.-J., & Chen T. (2000). Pose invariant face recognition. *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, pp.245-250.

MacQueen J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, CA, vol.1, pp.281-297.

Modha D. S., & Spangler W. S. (2003). Feature weighting in *k*-means clustering. *Machine Learning*, 52(3):

217-237.

Strehl A., Ghosh J., & Mooney R. J. (2000). Impact of similarity measures on web-page clustering. *Proceedings of AAAI2000 Workshop on AI for Web Search*, Austin, TX, pp.58-64.

Zhou Z.-H., Jiang Y., Yang Y.-B., & Chen S.-F. (2002). Lung cancer cell identification based on artificial neural network ensembles. *Artificial Intelligence in Medicine*, 24(1): 25-36.

Zhou Z.-H., Wu J., & Tang W. (2002a). Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 137(1-2): 239-263.

Appendix

Tables 5 to 9 present the detailed experimental results summarized in Section 4, where *single* denotes a single *k*-means clusterer, *w-voting* denotes *weighted-voting*, *sel-voting* denotes *selective voting*, *sel-w-voting* denotes *selective weighted-voting*, and *geometrical mean* denotes the average across all data sets. The 2nd to the 5th columns of the tables record the micro-p while the last column records how many clusterers have been selected by *selective voting* or *selective weighted-voting*. The values following '±' is the standard deviation.

Table 5. The clustering performance when ensemble size is 5. Note that after truncating from the 4th decimal digit, the differences on *waveform21* and *waveform40* are concealed.

data set	*single*	*voting*	*w-voting*	*sel-voting*	*sel-w-voting*	*selected*
image segmentation	.706 ± .021	.716 ± .038	.749 ± .040	.651 ± .087	.736 ± .040	2.00 ± .67
ionosphere	.722 ± .024	.711 ± .001	.768 ± .121	.768 ± .121	.768 ± .121	1.40 ± .52
iris	.878 ± .006	.887 ± .000	.887 ± .000	.862 ± .014	.862 ± .014	1.60 ± .52
liver disorder	.822 ± .007	.815 ± .007	.815 ± .007	.842 ± .016	.842 ± .016	2.20 ± 1.55
page blocks	.476 ± .027	.461 ± .058	.469 ± .063	.520 ± .046	.520 ± .051	1.70 ± .48
vehicle	.451 ± .018	.439 ± .019	.440 ± .024	.492 ± .055	.497 ± .053	1.40 ± .52
waveform21	.553 ± .000	.553 ± .000	.553 ± .000	.553 ± .000	.553 ± .000	1.70 ± .48
waveform40	.548 ± .000	.548 ± .000	.548 ± .000	.548 ± .000	.548 ± .000	1.90 ± .74
wine	.948 ± .006	.949 ± .005	.949 ± .005	.941 ± .031	.941 ± .031	1.50 ± .71
wpbc	.599 ± .002	.598 ± .000	.598 ± .000	.602 ± .009	.602 ± .009	4.20 ± 1.69
geometrical mean	.670 ± .011	.668 ± .013	.678 ± .026	.678 ± .038	.687 ± .034	1.96 ± 0.79

Table 6. The clustering performance when ensemble size is 8. Note that after truncating from the 4th decimal digit, the differences on *waveform21* are concealed.

data set	*single*	*voting*	*w-voting*	*sel-voting*	*sel-w-voting*	*selected*
image segmentation	.711 ± .017	.739 ± .060	.770 ± .051	.713 ± .051	.749 ± .055	3.20 ± 1.40
ionosphere	.721 ± .024	.711 ± .002	.768 ± .121	.768 ± .120	.768 ± .121	2.70 ± 0.95
iris	.879 ± .004	.891 ± .031	.904 ± .037	.850 ± .043	.862 ± .014	2.40 ± 0.70
liver disorder	.820 ± .005	.812 ± .001	.813 ± .005	.845 ± .012	.846 ± .012	2.70 ± 2.06
page blocks	.463 ± .020	.464 ± .058	.473 ± .060	.535 ± .029	.534 ± .035	2.70 ± 1.16
vehicle	.447 ± .012	.440 ± .018	.447 ± .036	.479 ± .034	.479 ± .039	2.30 ± 1.16
waveform21	.553 ± .000	.553 ± .000	.553 ± .000	.553 ± .000	.553 ± .000	2.70 ± 0.67
waveform40	.551 ± .009	.548 ± .000	.548 ± .000	.571 ± .071	.571 ± .071	3.50 ± 1.78
wine	.947 ± .004	.949 ± .002	.953 ± .008	.929 ± .040	.929 ± .040	2.30 ± 0.95
wpbc	.599 ± .002	.598 ± .000	.598 ± .000	.604 ± .010	.604 ± .010	6.00 ± 3.23
geometrical mean	.669 ± .010	.670 ± .017	.683 ± .032	.685 ± .041	.689 ± .040	3.05 ± 1.41

Table 7. The clustering performance when ensemble size is 13. Note that after truncating from the 4th decimal digit, the differences on *waveform21* are concealed.

data set	*single*	*voting*	*w-voting*	*sel-voting*	*sel-w-voting*	*selected*
image segmentation	.704 ± .013	.737 ± .023	.769 ± .038	.726 ± .047	.756 ± .058	5.30 ± 1.57
ionosphere	.717 ± .015	.710 ± .001	.767 ± .121	.769 ± .120	.769 ± .120	3.90 ± 1.66
iris	.877 ± .004	.886 ± .002	.886 ± .002	.808 ± .104	.829 ± .088	3.40 ± 1.26
liver disorder	.820 ± .004	.812 ± .000	.812 ± .000	.847 ± .006	.849 ± .002	3.30 ± 1.49
page blocks	.461 ± .014	.461 ± .056	.471 ± .061	.557 ± .024	.545 ± .004	4.10 ± 1.20
vehicle	.447 ± .013	.433 ± .002	.434 ± .006	.473 ± .030	.472 ± .037	3.20 ± 1.40
waveform21	.553 ± .000	.553 ± .000	.553 ± .000	.553 ± .000	.553 ± .000	4.70 ± 0.95
waveform40	.550 ± .006	.548 ± .000	.548 ± .000	.570 ± .072	.570 ± .072	4.80 ± 1.69
wine	.946 ± .003	.950 ± .010	.949 ± .010	.930 ± .040	.930 ± .040	4.00 ± 2.05
wpbc	.598 ± .002	.598 ± .000	.598 ± .000	.603 ± .018	.603 ± .018	7.10 ± 6.23
geometrical mean	.667 ± .007	.669 ± .010	.679 ± .024	.683 ± .046	.688 ± .044	4.38 ± 1.95

Table 8. The clustering performance when ensemble size is 20. Note that after truncating from the 4th decimal digit, the differences on *waveform21* are concealed.

data set	*single*	*voting*	*w-voting*	*sel-voting*	*sel-w-voting*	*selected*
image segmentation	.704 ± .009	.739 ± .040	.757 ± .044	.748 ± .059	.762 ± .062	8.60 ± 2.46
ionosphere	.719 ± .010	.710 ± .001	.852 ± .150	.852 ± .148	.853 ± .149	4.40 ± 3.63
iris	.876 ± .002	.879 ± .048	.912 ± .042	.829 ± .078	.826 ± .086	5.10 ± 1.20
liver disorder	.819 ± .003	.812 ± .000	.812 ± .000	.848 ± .003	.848 ± .003	4.40 ± 1.78
page blocks	.461 ± .010	.450 ± .042	.473 ± .059	.545 ± .005	.545 ± .004	7.00 ± 1.89
vehicle	.447 ± .008	.434 ± .003	.437 ± .010	.478 ± .015	.478 ± .017	5.30 ± 1.70
waveform21	.553 ± .000	.553 ± .000	.553 ± .000	.553 ± .000	.553 ± .000	8.10 ± 1.45
waveform40	.549 ± .004	.548 ± .000	.548 ± .000	.570 ± .072	.570 ± .072	7.70 ± 2.95
wine	.946 ± .003	.947 ± .007	.947 ± .008	.919 ± .046	.921 ± .043	5.80 ± 3.26
wpbc	.599 ± .002	.598 ± .000	.598 ± .000	.607 ± .019	.607 ± .019	7.20 ± 8.88
geometrical mean	.667 ± .005	.667 ± .014	.689 ± .031	.695 ± .045	.696 ± .046	6.36 ± 2.92

Table 9. The clustering performance when ensemble size is 30. Note that after truncating from the 4th decimal digit, the differences on *waveform21* are concealed.

data set	*single*	*voting*	*w-voting*	*sel-voting*	*sel-w-voting*	*selected*
image segmentation	.704 ± .008	.743 ± .048	.757 ± .055	.746 ± .076	.784 ± .067	13.00 ± 3.68
ionosphere	.720 ± .008	.710 ± .001	.910 ± .138	.909 ± .136	.911 ± .137	4.50 ± 5.06
iris	.877 ± .003	.858 ± .045	.910 ± .044	.853 ± .000	.853 ± .000	7.60 ± 2.37
liver disorder	.819 ± .003	.812 ± .000	.812 ± .000	.848 ± .003	.848 ± .003	6.40 ± 2.67
page blocks	.461 ± .009	.464 ± .057	.465 ± .054	.545 ± .005	.543 ± .002	10.30 ± 2.31
vehicle	.446 ± .007	.433 ± .003	.435 ± .005	.481 ± .010	.483 ± .013	7.60 ± 2.95
waveform21	.553 ± .000	.553 ± .000	.553 ± .000	.553 ± .000	.553 ± .000	12.90 ± 1.29
waveform40	.549 ± .002	.548 ± .000	.548 ± .000	.571 ± .071	.571 ± .071	10.80 ± 4.52
wine	.946 ± .003	.946 ± .006	.946 ± .007	.922 ± .041	.922 ± .041	9.00 ± 5.68
wpbc	.599 ± .001	.598 ± .000	.598 ± .000	.613 ± .020	.607 ± .025	2.00 ± 1.05
geometrical mean	.667 ± .004	.666 ± .016	.693 ± .030	.704 ± .036	.707 ± .036	8.41 ± 3.16