Boosting for Model-Based Data Clustering

Amir Saffari and Horst Bischof

Institute for Computer Graphics and Vision
Graz University of Technology, Austria
`saffari,bischof@icg.tugraz.at`

Abstract. In this paper a novel and generic approach for model-based data clustering in a boosting framework is presented. This method uses the forward stagewise additive modeling to learn the base clustering models. The experimental results on relatively large scale datasets and also Caltech4 object recognition set demonstrate how the performance of relatively simple and computationally efficient base clustering algorithms could be boosted using the proposed algorithm.

Key words: Boosting, Model-Based Clustering, Ensemble Methods.

1 Introduction

Currently the boosting methods are amongst the best techniques for classification problems. Recently, there has been a few attempts to bring the idea of boosting to the clustering domain [1–3]. In general, ensemble clustering refers to producing data partitions by utilizing results delivered by a set of clustering models. Such an ensemble could result in improvements in different aspects, such as performance, stability, and robustness, which are not attainable by individual models [4]. There are mainly two dominant topics of interest in ensemble clustering: 1) *consensus function* learning and 2) generating individual members of the ensemble. The consensus function learning [5–8, 4, 9] usually acts as a post-processing step to combine the results of different clustering models and usually does not deal with how the individual members of an ensemble are generated.

However, the main concern of this paper is the second major topic which deals with creating suitable models for the ensemble clustering tasks. In this context, Topchy et al. [2] nicely incorporated the idea of using a consistency index as a measure of how often a sample remains in the same cluster and preserves its label as the new models are introduced to the ensemble. The consistency index is then used for obtaining weights for the data samples. At each iteration, a new bootstrapped dataset is constructed by using the weights as probability distribution, and the next base model is trained over this dataset. The *k-means* algorithm is used as the base learning model. They have shown that this method results in a better performance compared to a non-boosting ensemble method [10] which uses a uniform sub-sampling of the dataset.

Frossyniotis et al. [1] also used the concept of sub-sampling the dataset, by using two different performance measures for assessing the clustering quality for each sample, both based on the membership values of each sample to the individual clusters. They incorporated a very similar approach used in the original Discrete AdaBoost [11] for updating the weights for both samples and base models. They compared the performance of the *k-means* and *fuzzy c-means* to their boosted versions and showed a better clustering results.

In this paper, we propose a novel boosting algorithm which provides a unified and general framework to include any model-based clustering method as its internal processing units. In Section 2, we will present this general boosting framework and use the concept of forward stagewise additive modeling [12] for finding solutions for ensemble members. The experimental results of the proposed methods on a few real world datasets will be demonstrated in Section 3.

2 Methods

Let $\chi = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \mathbf{x}_i \in \mathbb{R}^D$ be a finite set of samples. According to the bipartite graph view of model-based clustering [13], we define a clustering algorithm as a mapping from the input feature space, $\mathbb{R}^D$, to the space of data models which are usually described by probability density models. These density models usually are sampled from a known and fixed class of parametric functions. Therefore, we assume that in a clustering model, there is a collection of K probability density functions:

$$\mathbf{c}(\mathbf{x}) = [p_1(\mathbf{x}|\theta_1), \ldots, p_K(\mathbf{x}|\theta_K)]^T \tag{1}$$

where θ_k are the parameters of k^{th} model. Additionally, each clustering system is accompanied with an assignment function, $h(\mathbf{x}, k) = P(k|\mathbf{x})$, which based on $\mathbf{c}(\mathbf{x})$ determines the membership of a sample to k^{th} model. Therefore, we refer to the combination of the data models and the assignment function, $\mathcal{C}(\mathbf{x}) = \{\mathbf{c}(\mathbf{x}), h(\mathbf{x}, k)\}$ as a *clustering model*.

Note that Zhong and Ghosh [13] proposed this formulation as a general and unified framework for model-based clustering algorithms, which encompasses many different algorithms ranging from traditional *k-means* and *mixture of models* to *deterministic annealing*. For example in this framework, the *k-means* algorithm uses equivariant spherical Gaussian density models together with the following hard assignment function [14]:

$$h(\mathbf{x}, k) = \begin{cases} 1 & \text{if } k = \arg\max_y p_y(\mathbf{x}|\theta_y) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The parameter estimation procedure for the density models are usually conducted by maximizing the expected log-likelihood objective function:

$$\mathcal{L}(\theta, \mathcal{X}) = \sum_{n=1}^{N} P(\mathbf{x}_n) \sum_{k=1}^{K} h(\mathbf{x}, k) \log p_k(\mathbf{x}_n|\theta_k) \tag{3}$$

where $\theta = [\theta_1, \ldots, \theta_K]$ is the collection of all model parameters. In practice, the sample prior probability is unknown and it is common to set it to be a uniform distribution as: $\forall n : P(\mathbf{x}_n) = \frac{1}{N}$. Since some of the clustering algorithms do not directly address the probabilistic models, the objective function of Eq.(3) can be formulated in terms of a general loss function as:

$$\mathcal{L}(\theta, \mathcal{X}) = -\sum_{n=1}^{N} P(\mathbf{x}_n) \sum_{k=1}^{K} h(\mathbf{x}, k) d(\mathbf{x}_n, k) \tag{4}$$

where $d(\mathbf{x}_n, k)$ is the loss for the sample $\mathbf{x}_n$ under the k^{th} model (or cluster). For example, probabilistic models use $d(\mathbf{x}, k) = -\log p_k(\mathbf{x}|\theta_k)$, while in *k-means* this is usually expressed by $d(\mathbf{x}, k) = \|\mathbf{x} - \mu_k\|^2$ where μ_k is the k^{th} cluster center. We further simplify this objective function by introducing the sample loss function as $l(\mathbf{x}) = \sum_{k=1}^{K} h(\mathbf{x}, k) d(\mathbf{x}, k)$ and rewriting the overall objective function of Eq.(4) to be *minimized* as:

$$\mathcal{L}(\theta, \mathcal{X}) = \sum_{n=1}^{N} P(\mathbf{x}_n) l(\mathbf{x}_n) \tag{5}$$

2.1 Additive Modeling

Assume that a set of M different clustering models $\{\mathcal{C}_1, \ldots, \mathcal{C}_M\}$, are trained on a given dataset with equal number of partitions, K. We refer to each of these clustering models, $\mathcal{C}_m$, as the *base models* and to the collections of them as the *ensemble model*. The additive modeling approach is a general method for generating an aggregated model out of a set of base models. For example, the traditional boosting [11], and majority (plurality) voting strategies [15], can be seen as special cases of additive modeling [12].

In the context of data clustering, once the cluster correspondence between different models are solved, the assignment function for the ensemble can be represented in an additive form as:

$$H(\mathbf{x}, k) = \frac{1}{M} \sum_{m=1}^{M} h_m(\mathbf{x}, k) \tag{6}$$

where the function $H(\mathbf{x}, k)$ is an additive construction of M base functions, $\{h_m(\mathbf{x}, k)\}_{m=1}^{M}$. This is equivalent to the traditional scheme of classifier combination in the field of supervised learning problems. The corresponding objective function for the whole ensemble can be written in additive form as:

$$\mathcal{L}^M(\theta, \mathcal{X}) = \frac{1}{M} \sum_{n=1}^{N} P(\mathbf{x}_n) \sum_{m=1}^{M} l_m(\mathbf{x}_n) = \frac{1}{M} \sum_{n=1}^{N} P(\mathbf{x}_n) L^M(\mathbf{x}_n) \tag{7}$$

where $l_m(\mathbf{x}_n)$ is the sample loss of $\mathbf{x}_n$ under m^{th} base clustering model, and $L^M(\mathbf{x}) = \sum_{m=1}^{M} l_m(\mathbf{x})$ is the sample loss function for the ensemble of M base clustering models. It should be noted that since we usually train a specific class of algorithms, *e.g. k-means*, over the same dataset, their individual losses are comparable.

In order to facilitate the derivation of the new boosting algorithm for data clustering, we use an exponential form of Eq.(7) as:

$$\begin{aligned}\mathcal{L}^M(\theta,\mathcal{X}) = \frac{1}{M}\sum_{n=1}^{N} P(\mathbf{x}_n)L^M(\mathbf{x}_n) &< \frac{1}{M}\exp(\sum_{n=1}^{N} P(\mathbf{x}_n)L^M(\mathbf{x}_n)) \\ &\leq \frac{1}{M}\sum_{n=1}^{N} P(\mathbf{x}_n)\exp(L^M(\mathbf{x}_n)) = \frac{1}{M}\sum_{n=1}^{N} P(\mathbf{x}_n)L_e^M(\mathbf{x}_n) \end{aligned} \quad (8)$$

where $L_e^M(\mathbf{x}) = \exp(L^M(\mathbf{x}))$. This derivation uses the fact that $\forall x : \exp(x) > x$, $\sum_{n=1}^{N} P(\mathbf{x}_n) = 1$, and Jensen's inequality for the convex exponential transformation. This shows that the exponential loss function provides an upper bound on the actual loss shown in Eq.(7).

2.2 Learning

The *forward stagewise additive modeling* [12] is a general approach for learning the base models of the additive formulation of a function. It is an iterative algorithm which at each stage, finds the best model which if added to the previous set of models would result in an improvement in performance. It adds this model to the ensemble and fixes its parameter and continues the search for the next best model.

We apply the forward stagewise additive modeling approach to learn the parameters of the base clustering models. Note that at the i^{th} iteration, the exponential ensemble loss function in Eq.(8) can be written as:

$$L_e^i(\mathbf{x}) = \exp(l_i(\mathbf{x}))\prod_{m=1}^{i-1}\exp(l_m(\mathbf{x})) = L_e^{i-1}(\mathbf{x})\exp(l_i(\mathbf{x})) = w(\mathbf{x})\exp(l_i(\mathbf{x})) \quad (9)$$

where $w(\mathbf{x}) = L_e^{i-1}(\mathbf{x})$ is usually referred in boosting community as the sample weight, due to the fact that its value reflects the loss of previous models. Now from Eq.(8), we can derive the objective function for the i^{th} stage of boosting as:

$$\mathcal{L}_e^i(\theta,\mathcal{X}) = \frac{1}{i}\sum_{n=1}^{N} P(\mathbf{x}_n)w(\mathbf{x}_n)\exp(l_i(\mathbf{x}_n)) \quad (10)$$

which is the weighted version of the exponential loss function. Since as base models we use normal model-based clustering algorithms, and they are designed to minimize the loss function represented in Eq.(5), not the exponential form of it, we show that under a mild condition the optimal solution of model for Eq.(5) can be also an optimal solution of Eq.(10). If we assume that the sample loss is sufficiently small, we can approximate the exponential term with the first order Taylor expansion as:

$$\mathcal{L}_e^i(\theta, \mathcal{X}) = \frac{1}{i}\sum_{n=1}^{N} P(\mathbf{x}_n)w(\mathbf{x}_n)\exp(l_i(\mathbf{x}_n)) \simeq \frac{1}{i}\sum_{n=1}^{N} P(\mathbf{x}_n)w(\mathbf{x}_n)(1 + l_i(\mathbf{x}_n))$$
$$= \frac{1}{i}\big(\sum_{n=1}^{N} P(\mathbf{x}_n)w(\mathbf{x}_n) + \sum_{n=1}^{N} P(\mathbf{x}_n)w(\mathbf{x}_n)l_i(\mathbf{x}_n)\big) \tag{11}$$

Note that the first term in Eq.(11) is a fixed constant, so the optimization process only depends on the last term which is exactly the weighted version of normal objective function shown in Eq.(4). Therefore, finding a solution for the following loss function:

$$\mathcal{L}^i(\theta, \mathcal{X}) = \frac{1}{i}\sum_{n=1}^{N} P(\mathbf{x}_n)w(\mathbf{x}_n)l_i(\mathbf{x}_n) \tag{12}$$

ensures the optimality for the weighted exponential loss function Eq.(10). Note that the assumption of $l_i(\mathbf{x})$ to be small can be ensured for all clustering models by scaling their original loss to a suitable range, which of course does not have any effect on the solution they deliver.

2.3 Weak Learners

In supervised boosting methods, the base models are sometimes called *weak learners* due to the fact that these models need to perform just better than random guessing. Because of lack of labels, obtaining a similar concept is usually difficult for clustering tasks. However, based on our boosting framework, we propose a condition over base models which can serve both as a definition of weak learner and also as an early stopping criteria for the boosting iterations.

In order to achieve an overall better clustering results, it is desirable that with each addition of a base model, the average loss does not increase compared to the previous setup of base models. In terms of loss function, this translates into the condition of $\mathcal{L}_e^i \leq \mathcal{L}_e^{i-1}$. If we use the Eq.(10) and Eq.(8), we can write this condition as:

$$\frac{1}{i}\sum_{n=1}^{N} P(\mathbf{x}_n)L_e^i(\mathbf{x}_n) \leq \frac{1}{i-1}\sum_{n=1}^{N} P(\mathbf{x}_n)L_e^{i-1}(\mathbf{x}_n)$$
$$\beta_i = \frac{\sum_{n=1}^{N} P(\mathbf{x}_n)w(\mathbf{x}_n)\exp(l_i(\mathbf{x}_n))}{\sum_{n=1}^{N} P(\mathbf{x}_n)w(\mathbf{x}_n)} \leq \frac{i}{i-1} \tag{13}$$

We use this condition during the boosting iterations to prevent addition of newer models which are not able to provide better solutions to the current state of the ensemble.

Algorithm 1 CBoost: Boosting for Model-Based Data Clustering

```
 1: Input dataset: $\chi = \{\mathbf{x}_n\}, n = 1, \dots, N$.
 2: Set: $w_n = 1/N, n = 1, \dots, N$.
 3: for $m = 1$ to $M$ do
 4:     Compute: $\theta_m = \arg\min_{\theta} \sum_{n=1}^{N} P(\mathbf{x}_n) w_n l(\mathbf{x}_n)$.
 5:     Compute: $\beta_m = \frac{\sum_{n=1}^{N} P(\mathbf{x}_n) w_n \exp(l_m(\mathbf{x}_n))}{\sum_{n=1}^{N} P(\mathbf{x}_n) w_n}$.
 6:     if $m > 1$ and $\beta_m > \frac{m}{m-1}$ then
 7:         Break.
 8:     end if
 9:     Set: $\forall n : w_n \leftarrow w_n \exp(l_m(\mathbf{x}_n))$
10:     Optional: Set: $\forall n : w_n \leftarrow \frac{w_n}{\sum_{n=1}^{N} w_n}$.
11: end for
12: Find the cluster correspondence or a suitable consensus function for the clustering
    models.
```

2.4 Discussion

The overall algorithm is shown in Algorithm.1. There are a few remarks regarding this method. First, it provides a unified and generic boosting framework which is able to include any model-based clustering algorithm as its building blocks. In fact, any clustering algorithm which inherently optimizes a loss function can be used here. Optionally, it is beneficial for the algorithms to be able to use the sample weights in their optimization procedures. For those methods which lack such ability, one can always use the bootstrap methods to reflect the effect of sample weights.

Additionally, in this framework, there is essentially no need for solving cluster correspondence during the boosting iterations. In fact, after boosting training is over, it is quite possible to benefit from the state-of-the-art consensus function learning methods to derive the final clusters out of the ensemble.

Finally, the computational complexity of this algorithm is mainly centered around the complexity of its base models, as the only over-head operations are just a simple update of sample weights together with a check for early stopping condition.

3 Experiments

3.1 Methodology

As base models, we use *ETree* [16], and *k-means*. The *ETree* has a fast training and indexing capabilities with a complexity of $\mathcal{O}(N \log N)$ [16], however in general it provides inferior results compared to *k-means*.

In order to introduce the sample weights into the *ETree* algorithm, the update rate of the leaf nodes for each sample is multiplied with the $\eta_n = \frac{1}{1+\exp(-w_n^{'})}$, $w_n^{'} = \frac{w_n - \mu_w}{\sigma w}$. Since *ETree* does not deliver a fixed set of clusters, we apply an

Table 1. Datasets used in the experiments.

Dataset	No. Classes	No. Features	No. Samples
Synth	6	60	600
Pendigit	10	16	7494
Isolet	26	671	6238
Caltech4	4	420	1200

agglomerative clustering with group linkage over the leaf nodes to produce the desired number of clusters. For the *k-means* we use the same algorithm as in [3], which uses the weighted average for calculating the location of cluster centers.

To assess the quality of the clustering results, because *ETree* does not directly address a particular density model, it is not possible to measure log-likelihood for this model. Therefore, we use the Normalized Mutual Information (*NMI*) [5] between the true labels and the labels returned by the clustering algorithms. Besides its statistical meaning, the normalized mutual information enables us to compare clustering models with different number of partitions. Additionally,

3.2 Datasets

In order to show the performance of the proposed methods, we use four real-world datasets, which are summarized in Table 1. The Synthetic Control, Pendigit, and Isolet datasets are obtained from UCI machine learning [17] and UCI KDD repositories [18]. These datasets have been standardized before performing any clustering over them, i.e. for each feature, the mean has been subtracted and then feature values have been divided by their standard deviation. Additionally, we created a dataset from Caltech 4 object categorization repository by randomly selecting 400 images from each category of Airplanes, Cars, Faces, and Motorbikes, and then applying the Pyramid of Histograms of Orientation Gradients (PHOG) [19] shape descriptor to these images. We used a 2-level pyramid with 20-bin histograms using the same settings proposed by Bosch et al. [19]. We normalize these descriptors to have a unit L_1-norm.

3.3 Results

Table 2 and Table 3 show the performance of *ETree* and *k-means* respectively, together with their boosted versions over different number of cluster centers. The values are the average and standard deviation of NMI over 50 independent runs of the algorithms. We also implemented the algorithms proposed in [1, 2] and compared their performance with our method. Additionally, we applied different consensus function learning methods, namely voting [20], HGPA, and MCLA [5], to the final ensemble. For all boosting methods we set the maximum number of base models to be 50 for *ETree* and 20 for *k-means*, respectively. Since the method presented in [2] does not have an early stopping criteria, we report its best result over different number of base models.

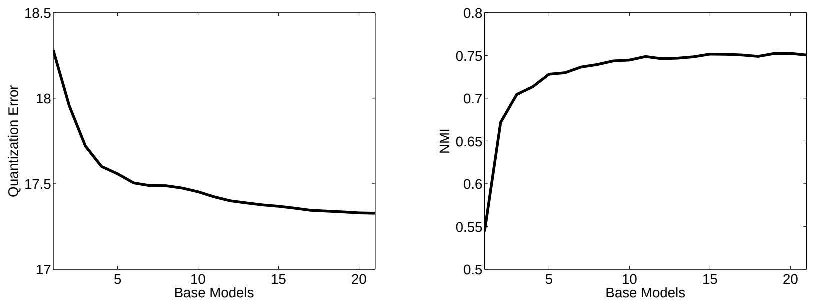
Table 2. Results of clustering using *ETree* in terms of $\%NMI$. K is the number of experimental cluster centers. The next four columns are the performance of *ETree* itself together with using it as the base model of our boosting algorithm (CB) and applying voting [20], HGPA, and MCLA [5] consensus function learning methods. The next two columns indicate the results of using the methods of Frossyniotis et al. [1] and Topchy et al. [2] with *ETree*. For those methods we only report their best performance over applying different consensus functions.

Dataset	K	ETree	CB.Vote	CB.HGPA	CB.MCLA	Fross [1]	Topchy [2]
Synth	3	72 ± 06	75 ± 01	75 ± 03	75 ± 01	71 ± 03	68 ± 01
	6	67 ± 04	80 ± 03	79 ± 01	83 ± 03	68 ± 03	75 ± 02
	12	69 ± 03	78 ± 02	75 ± 03	81 ± 01	70 ± 04	73 ± 02
Pendigit	5	57 ± 05	59 ± 03	58 ± 01	59 ± 01	55 ± 04	54 ± 02
	10	66 ± 03	69 ± 02	68 ± 03	69 ± 02	66 ± 03	69 ± 03
	20	68 ± 02	73 ± 01	73 ± 02	73 ± 02	70 ± 06	69 ± 01
Isolet	13	52 ± 03	71 ± 02	67 ± 02	72 ± 02	70 ± 02	71 ± 02
	26	56 ± 03	75 ± 01	69 ± 03	76 ± 03	68 ± 03	72 ± 04
	39	58 ± 01	73 ± 01	66 ± 02	73 ± 01	70 ± 05	71 ± 03
Caltech	4	41 ± 02	45 ± 03	43 ± 01	46 ± 01	42 ± 01	43 ± 02
	20	45 ± 05	48 ± 02	44 ± 03	48 ± 01	48 ± 03	47 ± 01
	40	42 ± 03	46 ± 02	45 ± 03	47 ± 02	41 ± 03	43 ± 03

Table 3. Results of clustering using *k-means* and its boosted form. The details are exactly the same as Table 2.

Dataset	K	k-means	CB.Vote	CB.HGPA	CB.MCLA	Fross [1]	Topchy [2]
Synth	3	70 ± 09	77 ± 04	77 ± 02	78 ± 02	72 ± 03	73 ± 08
	6	75 ± 03	78 ± 02	76 ± 03	82 ± 02	77 ± 03	76 ± 03
	12	76 ± 05	78 ± 02	77 ± 01	80 ± 03	80 ± 04	76 ± 03
Pendigit	5	55 ± 02	55 ± 03	54 ± 02	57 ± 03	56 ± 04	54 ± 04
	10	68 ± 03	69 ± 01	64 ± 03	69 ± 02	66 ± 03	67 ± 03
	20	72 ± 02	74 ± 02	72 ± 02	74 ± 03	65 ± 06	70 ± 04
Isolet	13	65 ± 04	69 ± 01	67 ± 02	71 ± 03	68 ± 02	66 ± 01
	26	69 ± 02	71 ± 02	70 ± 03	72 ± 02	70 ± 03	71 ± 03
	39	70 ± 03	71 ± 02	69 ± 02	72 ± 01	70 ± 05	70 ± 01
Caltech	4	37 ± 05	38 ± 03	37 ± 02	39 ± 02	39 ± 02	37 ± 02
	20	46 ± 04	46 ± 04	43 ± 03	48 ± 01	45 ± 03	47 ± 02
	40	47 ± 06	48 ± 01	47 ± 02	49 ± 01	47 ± 02	46 ± 03

From these results, it seems that in single model track, the *k-means* method has an expected advantage over *ETree* clustering. However, it should be noted that in our C++ implementation of both algorithms, *ETree* is about 15 times faster than *k-means*. Now looking into the results of our boosting algorithm, it is obvious that in most cases, the performance of the boosted base models are improved compared to the individual models. This clearly shows the advantage of incorporating a boosting framework for clustering algorithms. Additionally, con-



(a) Quantization Error (b) NMI

Fig. 1. Average vector quantization error (a) and NMI (b) versus the number of boosting iterations for the Isolet dataset using ETree as the base model.

sidering the variance of the results, the stability and robustness of the clusterings are also improved in most cases.

Looking further into these tables, it becomes clear that the boosted version of *ETree* seems to be as competitive as boosted *k-means*. It emphasizes the power of boosting method to transform a weaker method into a strong one. This is also a very desirable property as in large scale datasets, the *ETree* outperforms the *k-means*, in terms of computation time during both training and query phases.

Comparing the performance of different consensus function learning methods, we can see that the MCLA is performing slightly better than voting, and the HGPA seems to be not as competitive. Also we can see that in most cases our method outperforms the previous algorithms proposed in [1, 2].

Figure 1 shows the average quantization error (the loss which both *ETree* and *k-means* are optimizing) and the average NMI values, with respect to the addition of each base model during the boosting iterations for the Isolet dataset using the *ETree* as the base model. It is obvious that on average, with each boosting iteration and addition of a new base model the average error is decreasing, while the average NMI is gradually increasing. This demonstrates the success of the overall optimization process of the additive model proposed for the clustering tasks. Additionally, in most experiments, the convergence of the quantization error found to be an indication of the convergence for the NMI values. This experimental observation suggest the effectiveness of the early stopping method based on the convergence of the average loss during the boosting iterations.

4 Conclusion

In this paper, a novel approach for clustering datasets in a boosting framework was presented. The proposed methods are based on theoretical concepts of optimization using additive models in a forward stagewise approach. The experimental results, on a few real world and relatively large scale datasets demonstrate the benefits of using an ensemble of base models in the proposed boosting framework.

Acknowledgment

This work has been supported by the FWF Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04.

References

1. Frossyniotis, D., Likas, A., Stafylopatis, A.: A clustering method based on boosting. Pattern Recognition Letters **25** (2004) 641–654
2. Topchy, A., Bidgoli, M.B., Jain, A.K., Punch, W.F.: Adaptive clustering ensembles. In: Proc. of ICPR. Volume 1. (2004) 272–275 Vol.1
3. Nock, R., Nielsen, F.: On weighting clustering. IEEE Trans. PAMI **28**(8) (2006) 1223–1235
4. Topchy, A., Jain, A.K., Punch, W.: Clustering ensembles: Models of consensus and weak partitions. IEEE Trans. PAMI **27**(12) (2005) 1866–1881
5. Strehl, A., Ghosh, J.: Cluster ensembles-a knowledge reuse framework for combining multiple partitions. JMLR **3** (2002) 583–617
6. Fred, A., Jain, A.K.: Data clustering using evidence accumulation. In: Proc. of ICPR. (2002) 276–280
7. Dudoit, S., Fridlyand, J.: Bagging to improve the accuracy of a clustering procedure. Bioinformatics **19**(9) (2003) 1090–1099
8. Fischer, B., Buhmann, J.M.: Path-based clustering for grouping of smooth curves and texture segmentation. IEEE Trans. PAMI **25**(4) (2003) 513–518
9. Viswanath, P., Jayasurya, K.: A fast and efficient ensemble clustering method. In: Proc. of ICPR. (2006) 720–723
10. Bidgoli, M.B., Topchy, A., Punch, W.F.: Ensembles of partitions via data resampling. In: Proc. of ITCC. Volume 2. (2004) 188–192 Vol.2
11. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Proc. of ICML. (1996) 148–156
12. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. The Annals of Statistics **38**(2) (2000) 337–374
13. Zhong, S., Ghosh, J.: A unified framework for model-based clustering. JMLR **4** (2003) 1001–1037
14. Kearns, M., Mansour, Y., Ng, A.Y.: An information-theoretic analysis of hard and soft assignment methods for clustering. In: Proc. of UAI. (1997) 282–293
15. Matan, O.: On voting ensembles of classifiers. In: Proc. of the AAAI-96 Workshop on Integrating Multiple Learned Models. (1996) 84–88
16. Pakkanen, J., Iivarinen, J., Oja, E.: The evolving tree-analysis and applications. IEEE Trans. NN **17**(3) (2006) 591–603
17. Blake, Merz, C.J.: UCI repository of machine learning databases (1998)
18. Hettich, S., Bay, S.D.: The UCI KDD archive (1999)
19. Bosch, A., Zisserman, A., Munoz, X.: Representing shape with a spatial pyramid kernel. In: Proc. of CIVR. (2007) 401–408
20. Topchy, A.P., Law, M.H.C., Jain, A.K., Fred, A.L.: Analysis of consensus partition in cluster ensemble. In: Proc. of ICDM. (2004) 225–232