A FAMILY OF HIERARCHICAL CLUSTERING ALGORITHMS BASED ON HIGH-ORDER DISSIMILARITIES

Helena Aidos and Ana Fred

Instituto de Telecomunicações, Instituto Superior Técnico, Lisbon, Portugal

ABSTRACT

Traditional hierarchical techniques are used in many areas of research. However, they require the user to set the number of clusters or use some external criterion to find them. Also, they are unable to identify varying internal structures in classes, *i.e.* classes can be represented as unions of clusters. To overcome these issues, we propose a family of agglomerative hierarchical methods, which integrates a high-order dissimilarity measure, called dissimilarity increments, in traditional linkage algorithms. Dissimilarity increments are a measure over triplets of nearest neighbors. This family of algorithms is able to automatically find the number of clusters using a minimum description length criterion based on the dissimilarity increments distribution. Moreover, each algorithm of the proposed family is able to find classes as unions of clusters, leading to the identification of internal structures of classes. Experimental results show that any algorithm from the proposed family outperforms the traditional ones.

Index Terms— Hierarchical clustering, dissimilarity increments, agglomerative methods

1. INTRODUCTION

Clustering techniques organize patterns into groups or clusters, with no prior information about pattern labeling. That assignment is such that patterns belonging to the same cluster are similar, according to some proximity measure. Many clustering techniques have been developed, each one addressing differently issues such as cluster shape, number of clusters, and so on. In the literature, clustering techniques can be found in artificial intelligence, biology, image processing, marketing and many other areas [1, 2].

Several strategies can be adopted, however there are two major strategies found in the literature: partitional and hierarchical methods [1, 3, 4]. Partitional methods assign each pattern to a single cluster, and the number of clusters is set beforehand as a design parameter. The most typical and used partitional algorithm is k-means, which is a prototype-based method [1, 5]. Examples of other algorithms are probabilistic approaches, which assume that the data come from a mixture of models whose distributions are to be learned [1, 2].

Hierarchical methods produce a set of nested partitions in a hierarchical structure according to a proximity measure. Two main categories can be stressed out: agglomerative and divisive methods. Hierarchical methods belonging to the first category start by considering each pattern as a single cluster, and each partition is obtained from the previous one by merging two clusters into a single cluster, according to a proximity criterion. Single-link (SL) and average-link (AL) are two of the most used methods from this category [5]. Divisive methods start with a single cluster gathering all patterns and a divisive procedure is applied repeatedly until all clusters are singletons. This type of methods are very expensive computationally, for a cluster with N objects, there are $2^{N-1} - 1$ possible divisions [1, 2].

As mentioned above, patterns are assigned to a cluster according to some proximity measure. The choice of such measure can be difficult, since no prior information about the cluster shapes or structure is available. Most of the clustering techniques found in the literature use pairwise distances between patterns to perform that assignment, being the Euclidean distance the most typical one. A high-order dissimilarity measure has been proposed [6], the *dissimilarity increments*, consisting of a measure over triplets of nearest neighbor patterns. Moreover, the distribution of such measure has been derived under the hypothesis of local Gaussian generative models for the data [7].

In [8] an agglomerative hierarchical clustering algorithm was proposed based on the dissimilarity increments distribution (DID) following a SL strategy. In this paper, we propose a hierarchical family of clustering algorithms based on DID, which integrates DID in the traditional hierarchical clustering algorithms, namely SL, AL, complete-link (CL) and Ward's linkage (WL). These new methods are suitable to identify the structure of a class, *i.e.*, each class is defined as unions of one or more clusters, where each cluster follows the DID. Thus, we compare each baseline hierarchical algorithm with its DID-based extended version: SL vs SLDID, AL vs ALDID, CL vs CLDID, and WL vs WLDID.

This paper is organized as follows: section 2 presents the dissimilarity increments definition and its distribution. The proposed family of agglomerative hierarchical clustering is

This work was supported by the Portuguese Foundation for Science and Technology grant PTDC/EEI-SII/2312/2012.

explained in section 3. Experimental results are presented in section 4 and conclusions are drawn in section 5.

2. HIGH-ORDER DISSIMILARITY

Assuming that $\mathbf{x}_i$ is a pattern from a dataset X, and $d(\cdot,\cdot)$ is some dissimilarity measure between patterns, such as Euclidean distance, a triplet of nearest neighbors $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ is obtained as follows:

$$\begin{aligned}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) - \text{ nearest neighbors}&\\ \mathbf{x}_j : j = \arg\min_l\{d(\mathbf{x}_l, \mathbf{x}_i), l \neq i\}&\\ \mathbf{x}_k : k = \arg\min_l\{d(\mathbf{x}_l, \mathbf{x}_j), l \neq i, l \neq j\}.&\end{aligned}$$

The **dissimilarity increment** [6] associated with the triplet is defined as

$$d_{\text{inc}}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) = |d(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{x}_j, \mathbf{x}_k)|\,. \tag{1}$$

The dissimilarity increment measure is useful to explore the structure of a cluster, due to the fact that it can identify and characterize sparse clusters in data. This identification is possible because dissimilarity increments between neighboring patterns should not occur with abrupt changes, and between well separated clusters will have higher values. Thus, dissimilarity increments between patterns in different clusters are positioned on the tail of the distribution of dissimilarity increments associated with a cluster. In that sense, Fred and Leitão [6] have extended the concept of dissimilarity increments between patterns, to the gap of a cluster. This new concept is particularly useful to compare clusters and understand if two groups form a larger cluster or not.

Assume that $\mathbf{x}_i$ and $\mathbf{x}_j$ are the closest pair of patterns such that $\mathbf{x}_i \in C_i$ and $\mathbf{x}_j \in C_j$, and $\mathbf{x}_k$ is the nearest neighbor of $\mathbf{x}_i$ within C_i and $d_t(C_i) \equiv d(\mathbf{x}_i, \mathbf{x}_k)$. We define **gap of a cluster C_i with respect to a cluster C_j**, $\text{gap}_{C_i}(C_j)$, as the asymmetric increase in the dissimilarity value given by

$$\text{gap}_{C_i}(C_j) = |d(\mathbf{x}_i, \mathbf{x}_j) - d_t(C_i)|\,. \tag{2}$$

The **dissimilarity increments distribution** (DID) was derived in [7], using the Euclidean distance as the dissimilarity measure $d(\cdot,\cdot)$, under the hypothesis of Gaussian distribution of data. This distribution was written as a function of the mean value of the dissimilarity increments, which is denoted as λ. The probability density function (pdf) of the dissimilarity increments is given by

$$\begin{aligned}p_{d_{\text{inc}}}(w;\lambda) = \frac{\pi\beta^2}{4\lambda^2} w \exp\left(-\frac{\pi\beta^2}{4\lambda^2}w^2\right) + \frac{\pi^2\beta^3}{8\sqrt{2}\lambda^3}\times&\\ \times\left(\frac{4\lambda^2}{\pi\beta^2} - w^2\right)\exp\left(-\frac{\pi\beta^2}{8\lambda^2}w^2\right)\text{erfc}\left(\frac{\sqrt{\pi}\beta}{2\sqrt{2}\lambda}w\right),&\end{aligned} \tag{3}$$

where $\text{erfc}(\cdot)$ is the complementary error function, and $\beta = 2 - \sqrt{2}$.

3. HIERARCHICAL CLUSTERING BASED ON DID

This section presents a family of agglomerative hierarchical methods, called Hierarchical Clustering based on Dissimilarity Increments Distribution (HCDID). The idea is to combine a linkage algorithm with the concepts presented in section 2. An agglomerative hierarchical method starts with each point in a cluster and progressively join pairs of clusters. HCDID starts in the same way, each point is a separate cluster, and the candidates to merge are chosen as being the most similar pair of clusters, *i.e.*,

$$\text{minDist} = \min\{d(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i \in C_i, \mathbf{x}_j \in C_j\}. \tag{4}$$

The decision whether two clusters should or not be merged is based on one of the following tests:

- Two clusters are automatically merged if both have less than M patterns, according to some merging function.
- Consider that C_j has less than M patterns and C_i has M or more patterns, then it is checked if the mean of the increments of C_j is smaller than α times the mean of increments of C_i, *i.e.* the increments of C_j fall in the tail of the DID of C_i. If it does not fall in the tail, clusters C_i and C_j are merged; otherwise, are kept separate.
- Now, consider that C_i and C_j have already M or more patterns each. So, it is checked if $\text{gap}_{C_i}(C_j)$ is in the tail of the DID of cluster C_i. When that happens, C_i is "frozen", meaning that C_i is no longer available for merging with other clusters. Similarly, a test for C_j with respect to C_i is performed, but only if C_i was not previously "frozen". We only allow one cluster to be "frozen" in each iteration of the algorithm.
- Finally, if neither C_i nor C_j are "frozen", it is computed the description length [8] for each cluster as

  $$\begin{aligned}\text{DL}(C_i) = \frac{1}{2}(1 - \log(12)) + \log\lambda_i&\\ + \frac{1}{2}\log(I(\lambda_i)) - \log p(w;\lambda_i),&\end{aligned} \tag{5}$$

  where λ_i is the parameter of the DID (3) for cluster C_i, and we used the expected Fisher information $I(\lambda_i) \equiv -\mathbb{E}[\frac{\partial^2 \log p(w;\lambda_i)}{\partial\lambda_i^2}]$. Moreover, the description length for the cluster resulting of merging C_i and C_j, $C_i \cup C_j$, is computed in the same way, assuming that λ_{ij} is the parameter of the DID for cluster $C_i \cup C_j$. Now, if $\text{DL}(C_i \cup C_j)$ has a lower value than $\text{DL}(C_i) + \text{DL}(C_j)$ (description length of leaving the clusters separate), clusters C_i and C_j are merged and form a new cluster; otherwise, the clusters are left separated.

This procedure continues until all pairs of clusters have been tested. An outline of the whole procedure is in Algorithm 1.

Notice that a merging function for the clusters has not been defined. So far HCDID is a generic algorithm that only

Algorithm 1 Hierarchical clustering based on dissimilarity increments distribution (HCDID).

```
Require: data with $N$ samples
Require: parameters $M$ and $\alpha$
Require: merging function $d_*(C_a, C_b)$
 1: Start by assigning each pattern to a cluster
 2: repeat
 3:   Choose the most similar pair of clusters $(C_i, C_j)$ not
      yet tested, according to eq. (4)
 4:   if $|C_i| < M$ and $|C_j| < M$ then
 5:     Merge clusters $C_i, C_j$ into a new cluster $C_b$ using
        $d_*(C_a, C_b)$
 6:   end if
 7:   if $|C_i| \geq M$ and $|C_j| < M$ then
 8:     if $d_{\text{inc}}(C_j)$ is not in the tail of the pdf of $d_{\text{inc}}(C_i)$
        then
 9:       Merge clusters $C_i, C_j$ into a new cluster $C_b$ using
          $d_*(C_a, C_b)$
10:     else
11:       Do not merge $C_i, C_j$
12:     end if
13:   end if
14:   if $|C_i| \geq M$ and $|C_j| \geq M$ then
15:     Compute $\text{gap}_{C_i}(C_j)$ and $\text{gap}_{C_j}(C_i)$
16:     Compute $\text{DL}(C_i)$, $\text{DL}(C_j)$ and $\text{DL}(C_i \cup C_j)$
17:     if $\text{gap}_{C_i}(C_j)$ is in the tail of the pdf of $d_{\text{inc}}(C_i)$ then
18:       Freeze cluster $C_i$
19:     else if $\text{gap}_{C_j}(C_i)$ is in the tail of the pdf of $d_{\text{inc}}(C_j)$
        then
20:       Freeze cluster $C_j$
21:     else if $\text{DL}(C_i \cup C_j) \leq \text{DL}(C_i) + \text{DL}(C_j)$ then
22:       Merge clusters $C_i, C_j$ into a new cluster $C_b$ using
          $d_*(C_a, C_b)$
23:     else
24:       Do not merge $C_i, C_j$
25:     end if
26:   end if
27: until all pairs of clusters should not be merged
28: return data partition $\mathcal{P}$
```

makes decisions on whether two clusters should or not be merged, hence the designation family of agglomerative hierarchical methods.

Now, lets consider the new formed cluster, $C_b = C_i \cup C_j$, obtained by merging C_i and C_j, and C_a is one of the remaining clusters formed in previous steps. Also, lets consider $|C_i|$ and $|C_j|$ as the number of patterns in cluster C_i and C_j, respectively. We define $*$LDID algorithms by characterizing the merging function, according to the distance measure $d_*(C_a, C_b)$ between clusters ($*$ can be the letter 'S', 'A', 'C' or 'W'). More specifically, we have:

- SLDID

$$d_S(C_a, C_b) = \min\{d(C_i, C_a), d(C_j, C_a)\}; \quad (6)$$

- CLDID

$$d_C(C_a, C_b) = \max\{d(C_i, C_a), d(C_j, C_a)\}; \quad (7)$$

- ALDID

$$\begin{aligned} d_A(C_a, C_b) =& \frac{|C_i|}{|C_i| + |C_j|} d(C_i, C_a) \\ &+ \frac{|C_j|}{|C_i| + |C_j|} d(C_j, C_a); \end{aligned} \quad (8)$$

- WLDID

$$\begin{aligned} d_W(C_b, C_a) =& \frac{|C_i| + |C_a|}{|C_i| + |C_j| + |C_a|} d(C_i, C_a) \\ &+ \frac{|C_j| + |C_a|}{|C_i| + |C_j| + |C_a|} d(C_j, C_a) \\ &- \frac{|C_a|}{|C_i| + |C_j| + |C_a|} d(C_i, C_j). \end{aligned} \quad (9)$$

It should be noticed that any other distance measure between clusters can be used as merging function. In such case a new algorithm in this family is obtained.

4. EXPERIMENTAL RESULTS

A total of 36 real-world datasets from two repositories were used in the experiments. The majority of the datasets are from the UCI Machine Learning Repository[1], and only a few datasets are from the 20-Newsgroups database[2]. A summary of the datasets is presented in Table 1.

We intend to compare pairs of clustering algorithms, namely, SL vs SLDID, AL vs ALDID, CL vs CLDID, and WL vs WLDID. The set of four DID-based algorithms will be referred as HCDID, and the corresponding set of traditional hierarchical algorithms will be referred as THC.

HCDID has some parameters that need to be set. Typically, M is set to 5 patterns, since it is the minimum number of patterns to compute a rough estimate of DID, and α is set to 7, to ensure that very high increments are rejected.

All the algorithms were ran without knowing the true class labels. The number of clusters for THC is obtained applying the lifetime criterion [9], while HCDID found intrinsically the number of clusters automatically. However, the number of clusters found may not be the true one, since these algorithms are useful for finding internal structure in classes, *i.e.*, each class can be described as unions of one or more clusters. The quality of each partition $\mathcal{P}$ is assessed by the consistency index (CI) [10] (also known as accuracy), which is

[1]http://archive.ics.uci.edu/ml

[2]http://www.ai.mit.edu/people/jrennie/20Newsgroups/

Dataset	N	p	Nc	Dataset	N	p	Nc	Dataset	N	p	Nc
crabs	200	5	2	house-votes	232	16	2	ionosphere	351	34	2
iris	150	4	3	log-yeast	384	17	5	pima	768	8	2
auto-mpg	398	6	2	wine	178	13	3	80x	45	8	3
biomed	194	5	2	breast	683	9	2	chromo	1143	8	24
malaysia	291	8	20	glass	214	9	4	imox	192	8	4
kimia	216	4096	18	liver	345	6	2	mfeat-fac	2000	216	10
mfeat-fou	2000	76	10	mfeat-kar	2000	64	10	mfeat-pix	2000	240	10
mfeat-zer	2000	47	10	nist16	2000	256	10	sonar	208	60	2
soybean1	266	35	15	soybean2	136	35	4	diff300	300	10	3
same300	297	20	3	sim300	291	20	3	austra	690	15	2
derm	366	11	6	german	1000	18	2	heart	297	13	2
uci-image	2310	18	7	vehicle	846	16	4	wdbc	569	14	2

Table 1. Real-world datasets. **N** is the number of samples, **p** the dimension of the feature space and **Nc** the number of classes.

the percentage of agreement between $\mathcal{P}$ and the ground truth information.

Moreover, a matched consistency index (CI*) is used in HCDID. This index finds the best match between clusters and true classes, and computes the percentage of correctly clustered patterns if one represents the classes as the union of clusters, where each cluster can only be used for one class. To make a fair comparison, we applied the CI over the THC assuming that the true number of clusters is known, this is the same as applying CI*.

Figure 1 presents the CI* for each pair of comparison and Table 2 presents the overall results for CI and CI* indexes.

From Figure 1, we notice that HCDID is overall better than THC, and the most significant improvement occurs for SLDID comparing to SL. Moreover, SLDID is always equal or better than SL. On the other hand, WLDID is better than WL, but the improvement is smaller, there are a few datasets where WL performs better. For the remaining clustering comparisons, only three or four datasets have better results when THC is applied.

From Table 2, only WL performs better than WLDID, assuming that the true number of classes is unknown: WL has an average CI of 46.8%, winning 23 out of 36 datasets, and WLDID has an average CI of 36.7%, winning 12 out of 36 datasets. However, if we consider a class as a union of clusters and use CI* to measure the performance of WLDID, it has an average of 70.1% winning 28 out of 36 datasets. While WL, when the true number of clusters is known has an average CI of 63.4%, winning 7 out of 36 datasets.

SLDID is the algorithm with the most significant improvement compared to SL, since when it is better than SL its improvement is on average 21.2% against 0.3%, when we use CI*, and with an improvement on average 25.9% for SLDID, against 12.7% for SL, when we use CI. SLDID wins 23 out of 36 datasets with CI and 31 out of 36 datasets with CI*, against 11 and 4 out of 36 datasets for SL, respectively.

From Table 2 we observed that HCDID is better than THC

	CI			CI*		
Alg	**Mean**	**count**	**Mean Dif**	**Mean**	**count**	**Mean Dif**
SL	34.9%	11	12.7%	40.2%	4	0.3%
SLDID	47.5%	23	25.9%	58.4%	31	21.2%
AL	38.0%	14	29.0%	51.6%	4	11.6%
ALDID	40.9%	22	23.2%	66.0%	32	17.6%
CL	38.1%	15	23.8%	53.9%	3	8.4%
CLDID	39.6%	21	19.5%	66.5%	32	15.0%
WL	46.8%	23	24.4%	63.4%	7	10.8%
WLDID	36.7%	12	16.4%	70.1%	28	11.4%

Table 2. Pairwise comparison of clustering algorithms: SL vs SLDID, AL vs ALDID, CL vs CLDID, and WL vs WLDID. Mean consistency index (CI) and matched consistency index (CI*) for each algorithm, and number of datasets (count) with better CI and CI*. Mean Dif indicates that, when one algorithm wins, it is better on average x% than the other algorithm.

in terms of CI. This is shown more clearly when measuring with CI*. This indicates that HCDID is able to find some internal structure in classes, *i.e.*, the true classes can be represented as unions of one or more clusters.

5. CONCLUSIONS

A family of agglomerative hierarchical methods was proposed based on a high-order measure, the dissimilarity increments. This family of algorithms integrates the recently derived distribution for the dissimilarity increments (DID) in linkage algorithms. An advantage of these methods compared to the traditional linkage algorithms is that they are able to automatically find the number of clusters using a minimum description length criterion. In comparison, traditional algorithms require the user to set the number of clusters or use some external criterion to find them.

Each algorithm of the proposed family is able to find

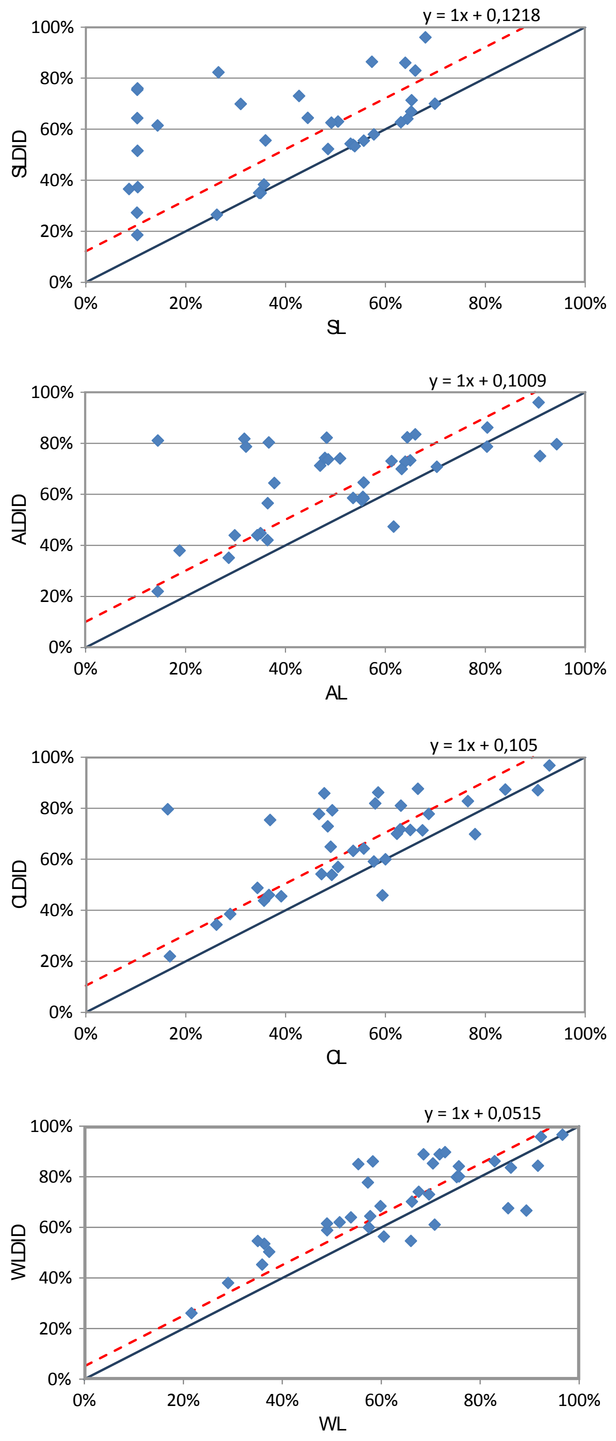

Fig. 1. Matched consistency index (CI*) comparing pairs of clustering algorithms. Dots represent datasets and the solid line, $y = x$, indicate equal CI* between the two algorithms. The dash line represents a linear regression line forced to be parallel to $y = x$, to indicate which algorithm is better (on average) and how much is the improvement.

classes as unions of one or more clusters, identifying internal structures in classes. This property leads to a significant improvement of teh performance of the algorithms compared to their corresponding traditional clustering algorithms.

REFERENCES

[1] Sergios Theodoridis and Konstantinos Koutroumbas, *Pattern Recognition*, Elsevier Academic Press, 4th edition, 2009.

[2] Brian S. Everitt, Sabine Landau, Morven Leese, and Daniel Stahl, *Cluster Analysis*, John Wiley & Sons Ltd., 5th edition, 2011.

[3] Anil K. Jain and Richard C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[4] Rui Xu and Donald Wunsch II, “Survey of clustering algorithms,” *IEEE Trans. on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[5] Anil K. Jain, M. Narasimha Murty, and Patrick Joseph Flynn, “Data clustering: a review,” *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[6] Ana Fred and José Leitão, “A new cluster isolation criterion based on dissimilarity increments,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 944–958, 2003.

[7] Helena Aidos and Ana Fred, “Statistical modeling of dissimilarity increments for d-dimensional data: Application in partitional clustering,” *Pattern Recognition*, vol. 45, no. 9, pp. 3061–3071, 2012.

[8] Helena Aidos and Ana Fred, “Hierarchical clustering with high order dissimilarities,” in *Proc. of the 7th Int. Conf. on Machine Learning and Data Mining (MLDM 2011)*, 2011, pp. 280–293.

[9] Ana Fred and Anil Jain, “Combining multiple clusterings using evidence accumulation,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 835–850, 2005.

[10] Ana Fred, “Finding consistent clusters in data partitions,” in *Proc. of the 2nd Int. Work. on Multiple Classifier Systems (MCS 2001)*, 2001, pp. 309–318.