Discussing a new Divisive Hierarchical Clustering algorithm

Erica Vidal, Pablo M. Granitto, and Ariel E. Bayá*

CIFASIS, French Argentine International Center for Information and Systems Sciences, UAM (France) / UNR-CONICET (Argentina)
Bv. 27 de Febrero 210 Bis, 2000, Rosario, Argentina
{granitto,baya}@cifasis-conicet.gov.ar
{ericavidal}@gmail.com

Abstract. We present DHClus, a new Divisive Hierarchical Clustering algorithm developed to detect clusters with arbitrary shapes. Our algorithm is able to solve clustering problems defined by different scales, i.e. clusters with arbitrarily dissimilar densities, connectivity or between cluster distances. The algorithm not only works under this difficult conditions but it is also able to find the number of clusters automatically. This paper describes this new algorithm and then present results on real gene expression data. We compare the results of DHClus with other algorithms to provide a reference frame.

1 Introduction

In this work we describe a new clustering algorithm which we named DHClus (**D**ivisive **H**ierarchical **Clus**tering). The algorithm presented here connects a some interesting ideas from the clustering literature with the aim to solve a number of relevant clustering problems. The clustering problems that we considered include clusters with arbitrary shapes defined by multiple scales. Also these clusters might be embedded in a high dimensional space. An example of multiple scales can be thought as a dataset containing a number of clusters but where each of the clusters is defined by a different density. In the case of three Gaussian components defined as $\{N(\mu_1, \sigma_1), N(\mu_2, \sigma_2), N(\mu_3, \sigma_3)\}$, where μ_i is the cluster centroid and σ_i^2 is the the cluster variance, the parameters $(\sigma_1, \sigma_2, \sigma_3)$ represent scales in terms of the density of each clusters. A different example of multiple scales could be thought in terms of highly dissimilar clusters separation or as a mix of scales, clusters with different densities separated by highly dissimilar distances. For the sake of simplicity, our previous example involved Gaussian clusters but the reader should keep in mind that DHClus is explicitly developed to manage non Gaussian data.

A major feature of DHClus is that it does not need a parameter specifying the number of clusters; rather, the algorithm by itself tries to find an optimal number

* Author to whom all correspondence should be addressed

of clusters. This feature is highly related with the ability of the algorithm to find clusters defined by different scales. Some key components of DHClus come from the use of ideas of Spectral Clustering[6]. In some sense DHClus is wrapped around a simplified version of Spectral Clustering. The type of problems that we have defined in the previous paragraph have been shown to be very difficult to solve by Spectral Clustering [7]. This new algorithm allow us to solve problems that the original algorithm cannot.

The rest of the paper is laid out as follows: Section 2 presents a description of DHClus and discusses a number of relevant issues that arise from analyzing the algorithm. Section 3 present results obtained from applying DHClus to five real world datasets. The section also includes results coming from other methods which provides a reference to evaluate the results from DHClus. Finally, Section 4 presents a summary of our findings and discuss some future work.

2 DHClus: A divisive hierarchical aproach for clustering.

DHClus is a clustering algorithm that finds the number of clusters in the data but considering that the clusters included in the dataset have different scales. The hierarchical structure of the algorithm is used to capture the different scales present in the data. The root of the tree has the complete dataset, then each new node of the tree is obtained by partitioning the data from its parent node in two. To partition the data contained in each parent node we look for two clusters with Spectral Clustering. We have followed the work of von Luxburg [6] and used the version of Spectral Clustering algorithm associated to the random walk that uses the Eigenvectors of the Laplacian matrix: $L = P^{-1}W$, where P is the degree matrix and W the similarity matrix. An important step in the clustering procedure is to calculate the similarity matrix, W, associated with the data. Most commonly used functions have parameters, which are dependent of the scale of the clusters. Since we are looking to solve problems with many different scales one possible solution would be to calculate the similarity matrix W considering all the scales involved. Instead we consider all the scales involved in the problem one at the time by using a tree and solving a subproblem with a single scale in each node.

An example is the Gaussian or RBF similarity, which can be defined as: $s(x_i, x_j, \sigma) = e^{\frac{-|x_i - x_j|^2}{2 \cdot \sigma^2}}$, where $|x_i - x_j|$ is the euclidean distance and σ is a parameter that depends on the scale. In this similarity the constant σ is related to the density of the data. If we were to use this similarity we would need to estimate σ and calculate W again for each scale present in the problem. We introduced this similarity as a simple example of what means to recalculate the scale parameters and similarity matrix (W) to consider all the scales present. In this work we have used a different similarity which will be described later.

Up to this point DHClus would simply divide the data until only singletons are left in each node. Such behavior would force us to look for the number of clusters by searching the tree. Instead we propose to validate the partition of each node, i.e. decide after we divide the data associated to a node if we accept the partition or if we stop and do not partition that node anymore. The algorithm stops when this criterion tell us that no more nodes need to be divided. Algorithm 1 describes DHClus using pseudo code but without much detail. In the following paragraphs we present a careful analysis of the algorithm.

input : $X \in \mathcal{R}^n$ a dataset, $s(,,\theta)$ a similarity function, θ_i an interval for the scale parameters.
output: $\{\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_{k_c}\}$ Clustering Labels, k_c number of clusters.
1 Optimize θ, $\underset{\theta}{\text{argmin}} = \psi(X, s(,,\theta), \theta = \theta_i)$;
2 Calculate W by using $s(,,\theta)$;
3 Divide X in two clusters X_1 and X_2 by partitioning W with Spectral Clustering;
4 Calculate label $\mathcal{L}_1$ and $\mathcal{L}_2$ corresponding to X_1 and X_2;
5 Validate X_1 and X_2, i.e. answer: do we have two clusters?;
6 **if** *two clusters* **then**
7 | DHClus$(X_1, s(,,\theta), \theta_i)$;
8 | DHClus$(X_2, s(,,\theta), \theta_i)$;
9 **else**
10 | return $\mathcal{L}_q$;
11 **end**

Algorithm 1: DHClus

Algorithm 1 inputs are the dataset X, which we would like to cluster, a similarity function $s(,,\theta)$ and θ_i an interval to look for the optimal values θ. The notation $s(,,\theta)$ implies that the parameters (θ) are intrinsic to the similarity function, which means that parameters are embedded inside the function. Also θ_i represents an arbitrary number of parameter. For each one a range of values is defined. There we will look for an optimum set of parameters. In this paper we use a similarity named RBF-PKNNG, which we define as $s_{pknng}(x_i, x_j, \theta) = e^{\frac{-d(x_i - x_j)^2_{pknng}}{2 \cdot \sigma^2}}$. The PKNNG metric, $d(x_i - x_j)_{pknng}$, was developed as a graph based metric which uses the concept of geodesic distance to calculate the distance between two points (x_i,x_j). This metric, developed by Bayá and Granitto [2], also uses penalized edges to join dense groups of points that are connected through low density regions of space. Due to space constrains we refer the reader to [2] for more information on the topic. The PKNNG metric was very useful to detect arbitrary shaped clusters in high dimensional data. The similarity s_{pknng} uses a parameter σ, which in this case has a different meaning that in the plain RBF similarity. The constant σ is related to the distances in the PKNNG graph such that the fractional quantity $\frac{d(x_i - x_j)^2_{pknng}}{2 \cdot \sigma^2}$ defines the meaning of close and far. There is also an extra parameter from the PKNNG

metric, k, which we need to consider. Finally, θ is defined as $\theta = \{\theta_k, \theta_\sigma\}$. There are two parameters that need to be set.

Line 1 from Algorithm 1 presents a function ψ that uses as input the data X, the similarity $s_{pknng}(,,\theta)$ and the interval θ_i, which accounts for k and σ. As a result, ψ finds a set of optimal values θ to set W using RBF-PKNNG, which then is used to divide the data with Spectral Clustering in two clusters. We present the following function as a candidate for ψ:

$$BH(k^*) = \frac{1}{k^*} \sum_{i=1}^{k^*} \frac{1}{n_i} \sum_{x \in C_i} ||x - m_i||^2. \quad (1)$$

where n_i is the number of elements in cluster C_i, k^* is the number of clusters and m_i is the centroid of cluster C_i. The function $BH(k^*)$ can be found in [1] as a clustering quality index. We use Equation (Eq.) 1 restricted to a single value, which is $BH(2)$ because we are looking for the best partition of two clusters. But Eq. 1 needs clustering assignments to find $BH(2)$ so we will need to calculate lines 1 to 4 as a block for each set of parameters. Equation 1 is not applied in the original space $\mathcal{R}^n$ where X is defined but in the space of the first and second Eigenvectors, which is where Spectral Clustering looks for two clusters. We are partitioning the data and looking for the optimal parameters (θ) at the same time. The notation: $\underset{\theta}{\text{argmin}} = \psi(X, s(,,\theta), \theta = \theta_i)$ means that we are looking for the minimum $BH(2)$ coming from two clusters defined in the space of the first and second Eigenvectors, which was generated by the parameters $\theta = \{\theta_k, \theta_\sigma\}$. At the end of the block made by lines 1 to 4, we are only required to retain the sets of clustering labels $\{\mathcal{L}_1, \mathcal{L}_2\}$ that have minimum $BH(2)$. This value, the minimum of $BH(2)$, represent the most compact representation of two clusters in the space of the first and second Eigenvectors.

To validate that X_1 and X_2 form two clusters, see Line 5, there are a number of methods that can be applied. This methods are usually referred in the literature as clustering validation [4]. We selected a variation of a well known test, The Gap Statistic [8], which is a weighed version of the test developed by Yan and Ye [9]. This new version of the test compensates for a number of problems in the original method. The issue with both versions is that they use the Euclidean metric to define the within sum of squares (WSS) and later perform a statistical test with it. For more information on these methods we refer

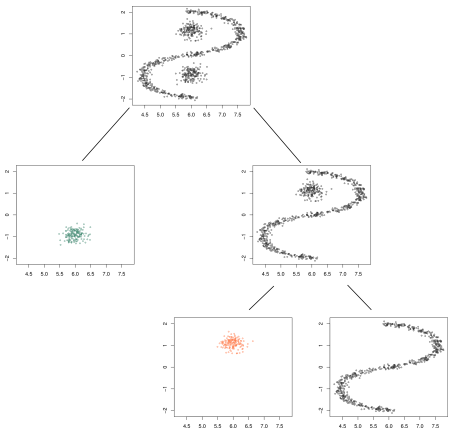

Fig. 1. Example of Algorithm 1 applied to the data shown in the root node of the tree.

the reader to [8,9]. It is known that
the WSS defined with the Euclidean metric cannot be used to validate non Gaussian data [3]. To solve this problem we change the usual definition of the WSS involving the Euclidean metric and use instead the PKNNG metric. The PKNNG metric is used with the parameter k found in lines 1 to 4. Then the new weighed Gap Test is simply applied assuming that there can be only one or two clusters, which answer our question: do we have two clusters?. A previous work from Bayá and Granitto [3] has shown good results using a different metric than the Euclidean to define the WSS. The results from this work motivated our choice of test in Algorithm 1.

Once the algorithm has decided if there are one or two clusters it can choose to continue and apply DHClus to a new pair of nodes associated to partition X_1 and X_2 by recursively calling DHClus (lines 7 and 8) or finish the process for the current node. Line 10 returns the labels corresponding to a leaf node or end node. Consistence for the labels is kept internally and the clustering labels $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_{k_c}\}$ are obtained with the labels returned in Line 10 and so are the number of cluster (k_c). Both $\mathcal{L}$ and k_c are the output of DHClus.

Figure 1 is the graphical equivalent of applying Algorithm 1 to the dataset associated to the root node of the tree. This dataset present both Gaussian and non Gaussian data. The difficulty of this dataset consists of finding clusters of different shapes and densities.

3 Results.

In this section we show some performance results of DHClus when it is applied to a set of gene expression datasets. Table 1 presents a brief description of these datasets. The datasets {Aml, Ali, Leu, Lung} are different types of human cancer cells while Yeast is a dataset of proteins. In the first set of examples we would like to cluster the types of cancers while in the Yeast dataset we would like to find functional groups of proteins. The relevance of these sets from a machine learning point of view comes from the high number of dimensions, the small number of samples and the different densities of the clusters. More detail of these datasets can be found in [2,3].

Our first concern is to verify if DHClus can solve the clustering problems defined by the datasets from Table 1, i.e. can DHClus find clusters with high degree of matching between these clusters and the golden rule. In real working conditions it is not usual to have the true labels of the data (or golden rule) but here we are using these datasets to estimate the behavior of our algorithm. Still this is not enough, we need to present our results and a reference to compare both. For this reason we contrast our results with Spectral Clustering (SP) using: the RBF similarity, the RBF-PKNNG similarity and Spectral Clustering using Local Scaling (LS) which is an improvement of the RBF similarity introduced to find

Dataset	Samples	Class & Sample distribution	Dimension
Aml	38	3 (11-8-19)	999
Ali	62	3 (42-9-11)	1000
Leu	248	6 (43-27-15-79-20-64)	985
Lung	197	4 (139-17-21-20)	1000
Yeast	208	4 (14-27-121-46)	79

Table 1. This table presents five gene expression datasets. The columns of the table describe important features of the dataset. The third column has the number of classes and the distribution of examples per class. The rest of them are well described by their names.

clusters with different densities. A full description of Local Scaling can be found in the work presented by Zelnik-Manor and Perona [10].

Dataset	Dhclus	SP_{rbf}	$SP_{rbf-pknng}$	SP_{ls}
Aml	1	1	0.91	1
Ali	1	1	1	1
Leu	0.92	0.75	0.55	0.68
Lung	0.64	0.91	0.65	0.82
Yeast	0.85	0.95	0.97	0.96

Table 2. Value of *cRand* for DHClus SP_{rbf}, $SP_{rbf-pknng}$ and SP_{ls}.

Table 2 presents the results of applying to each dataset: DHClus, Spectral Clustering with the RBF similarity (SP_{rbf}), Spectral Clustering with the RBF-PKNNG similarity ($SP_{rbf-pknng}$) and Spectral Clustering with Local Scaling (SP_{ls}). We use the Adjusted Rand Index (ARI or $cRand$)[5] to measure the degree of match between the known classes and the labels resulting from each clustering method. Table 2 presents the $cRand$ of a single run using all the data from each dataset. In the case of DHClus the algorithm itself is detecting the number of clusters and setting the internal parameters. On the other hand the three Spectral Clustering variants ($SP_{rbf}, SP_{rbf-pknng}, SP_{ls}$) need the number of clusters set by hand. Internal parameters in these methods are found in each case using Equation 1 in the space of the Eigenvectors like it was done with DHClus. For example in the case of LEU the parameters selected are those that minimize $BH(6)$ in the space of the first six Eigenvectors. Since Spectral Clustering is simultaneously looking for six clusters we select the parameters that lead to the most compact six clusters in the Eigenvectors space.

We performed a second experiment, which consist of analyzing the response of DHClus to a perturbation of the input data. We removed a fraction of 5 percent of the data and repeated the clustering with DHClus 100 times for each dataset. The aim of this experiment is to find out the degree of change in the output caused by a small perturbation of the input. These results are shown by Figure 2. Panel (a) presents the results using boxplots corresponding to the cRand measured in the (y) axis for each dataset, which is indicated in the (x) axis. The

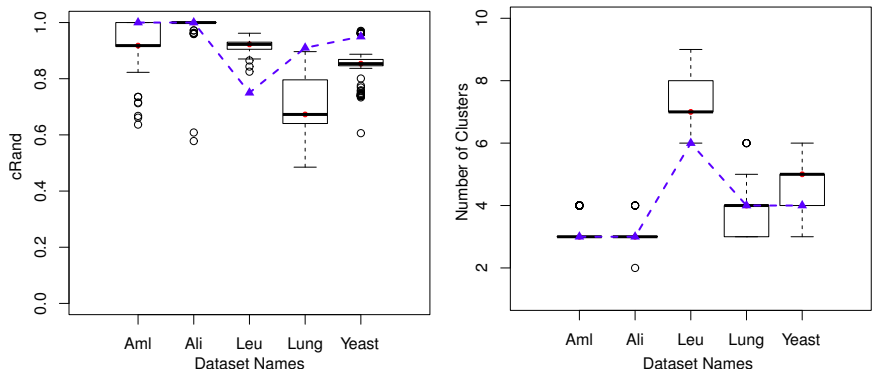

Fig. 2. Response to perturbation of DHClus. Panel (a) shows the cRand (y axis) for each real dataset (x axis). The blue line is the best solution of Spectral Clustering for each dataset. Panel (b) shows the number of clusters (y axis) for each real dataset (x axis). The blue line is the correct number of cluster extracted from Table 1.

blue line shows the best solution of Spectral Clustering for each dataset based on the single run results from Table 2. Panel (b) presents boxplots measuring the number of clusters obtained by DHClus for each dataset. The number of cluster can be read in the (y) axis and the dataset name in the (x) axis. In this case the blue line shows the number of classes of each dataset.

We direct our attention first to the results presented in Table 2. As we can see performances appear similar. DHClus finds a better clustering solution for Leu, a worst solution for Lung and a slightly worst one for Yeast. But one should not forget that DHClus is selecting both the number of clusters and internal parameters automatically while the SP variants need the information of the number of clusters to output the solution and to set any internal value. DHClus is finding similar results with less information. Also this kind of information needed by the SP variants is usually not available.

Figure 2.a shows the variation of the cRand index corresponding to a perturbation made by sub-sampling the input data. In both figures the result that stands out the most is the Lung dataset, which is actually the worst one. The problem with this example is that DHClus fails to recognize one class as a valid cluster. The results from Figure 2.b in the case of Lung show that many runs of DHClus find four clusters. A closer look to the output reveals that it is always finding three classes and when the method finds four clusters it is because one of the classes is being divided in two clusters. This example is the only one that fails to find one class. We found out too that sometimes the algorithm may over fragment the solution, i.e. divide one class in more clusters but only after recognizing correctly all classes. This is what happens for datasets Leu and Yeast but the number of clusters found is close to the number of classes.

4 Conclusions and future work.

We have presented a new Divisive Hierarchical Clustering algorithm, DHClus, which automatically finds the number of clusters and sets its internal parameters. Our results show that DHClus has similar performance than Spectral Clustering combined with different similarities. Our worst result was the Lung dataset and it happened when DHClus failed to find one of the classes. This kind of result is problematic since the solution tree was prematurely ended by the stop proposition (see Line 5 from Algorithm 1). This same proposition is also responsible for over fragmenting the solution but in such case it is possible to prune the solution tree to find a better clustering outcome. These errors occur because the stop proposition acts using only local information. It evaluates part of the data using a single scale. This scale used in the test is unique but each test can have a different scale. A pruning rule could improve performance by including global information using all scales and the data as a whole. Including a pruning function to DHClus is a future modification we intend to make. We also plan to look for other stopping criteria as well.

References

1. Ball, G.H., Hall, D.J.: Isodata. a novel method of data analysis and pattern classification. (1965)
2. Bayá, A.E., Granitto, P.M.: Clustering gene expression data with a penalized graph-based metric. BMC Bioinformatics 12(1), 2 (2011), http://www.biomedcentral.com/1471-2105/12/2
3. Bayá, A.E., Granitto, P.M.: How many clusters: A validation index for arbitrary-shaped clusters. IEEE/ACM Transactions on Computational Biology and Bioinformatics 10(2), 401–414 (2013)
4. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. J. Intell. Inf. Syst. 17(2-3), 107–145 (2001)
5. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1988)
6. Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4), 395–416 (2007), http://dx.doi.org/10.1007/s11222-007-9033-z
7. Nadler, B., Galun, M.: Fundamental limitations of spectral clustering. In: Advanced in Neural Information Processing Systems 19. MIT Press (2007)
8. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters in a dataset via the gap statistic. Journal of the Royal Statistical Society, Series B 63, 411–423 (2001)
9. Yan, M., Ye, K.: Determining the number of clusters using the weighted gap statistic. Biometrics 63(4), 1031–1037 (2007), http://dx.doi.org/10.1111/j.1541-0420.2007.00784.x
10. Zelnik-Manor, L., Perona, P.: Self-tuning spectral clustering. In: Advances in Neural Information Processing Systems 17. pp. 1601–1608. MIT Press (2004)