# Meta Clustering

Pedro Artigas, Anna Goldenberg, Anthon Likhodedov, Rich Caruana

## 1 Introduction

There is a constant need for comprehensive data representation. One of the useful methods to represent data is clustering. It decreases dimensionality presenting the user with a more comprehensible data portrayal. The current approach to clustering is to find "*the* single correct/optimal/best" model. The truth is that no expert will be satisfied with a particular single representation of data even though a sophisticated algorithm found that this specific model is the best fit for the data at hand. Even if the expert finds that the model is satisfactory for his purpose, another expert having to deal with the same data for a slightly different purpose may not find the model usable at all.

This project proposes a different approach to data analysis and representation. The idea is to generate many clustering models of the same dataset and present the expert with few qualitatively different ones.

It is important to note that it is not a simple task and there are several important issues that have to be resolved in order to establish the success of the given project: How to generate qualitatively different clusterings? When clusterings are generated, how can we show that they are really different? How should we select few final models from the whole family of different ones?

The mechanism for automation of the choice of final models is called metaclustering. It is based on the notion of similarity or dissimilarity of different clusterings: we define a function $f : C \times C \mapsto \mathbb{R}$, where $C$ is a space of clustering models (i.e. our hypothesis space). This function allows us to abstract from the complicated nature of the objects we are dealing with and work with a set of $f$ values (i.e. distances). Then, having a set of distances between different models we can apply one of the clustering algorithms and obtain a compact representation of the clusterings' set (few qualitatively different models).

The choice of $f$ is a very important issue. Another one is how to make sure that the selected clusterings are meaningful.

Answering all these questions, make this project a challenging task.

The rest of the report is structured in the following way: the "previous work" section provides a brief review of the current approaches to clustering, including the latest major advances in this field. We describe our approach and discuss the methods for generating different clustering models and some of their attractive properties in section 3. The choice of $f$ and metaclustering are described in section four. In the last section we review the results of applying our method to real "protein" dataset and a synthetic 2-dimensional dataset. Also the results of some experiments, justifying the choice of the method selected are provided.

There are several reasons why this particular project is a very interesting and open one. Since the current direction is in finding the "best" model, there is a good chance that the results of the project tested on a synthetic easily visualized dataset and confirmed by the expert on a domain specific dataset would be different from anything that exists up to this point. Since the project is not data specific, it may attract attention of scientists interested in understanding and classifying data.

## 2 Previous work

Partitioning a given set of points into homogeneous groups is one of the most fundamental problems in pattern recognition. It belongs to a class of unsupervised learning problems.

By analogy with the information theory concepts, clustering can be viewed as an optimization problem where we look for an optimal representation of a large dataset with a more compact "code" (set of clusters of data points). The functional to be minimized is referred to as distortion, measuring how good our representation is.

There exist two different forms of the problem setup dependent on the data representation. In the first form data points are assumed to have a vector form and clustering means deriving a set of vectors that quantize the data with minimal error. In the second setting, known as pairwised clustering, the data points are characterized by distances between them instead of coordinates, and the dataset structure is hidden in the distances' matrix.

Obviously the second setting is more difficult; however in many cases the set of pairwise distances is the only information available. Moreover sometimes the "distances" function is not a metric (e.g. the triangle inequality does not hold) that makes the task of finding a consistent low-dimensional representation of the data impossible.

Both settings were approached with several different methods and algorithms. The most popular algorithm for the first one is K-mean, belonging to a class of EM methods. It is based on the idea of iterative reestimation of optimal "codevectors" (centroids of clusters) interleaved with repartitioning of the dataset based on the new centroid values. The shortcomings of this algorithm are 1) poor scaling, 2)unknown number of clusters ($K$), 3) convergence to local minima - the iterative process may not find the globally optimal partitioning and 4)vector representation of data is required.

The problem of selection of $K$ is handled by ISODATA (Iterative Self Organizing Data Analysis Technique) algorithm. Its idea is in following rules for splitting or merging clusters, while repeating a clustering procedure after each such operation.

Avoiding the poor local minima in K-mean is achieved by careful selection of initial points. The algorithm for that is described in [8] by Bradley and Fayyad. It showed a 20% average decrease of the distortion over a random version of $K$-mean (where the initial points are chosen randomly) on a dataset from the Reuters text classification database.

The solutions for the first two problems are described by Andrew Moore and Dan Pelleg in [2]. They provide a fast algorithm for $K$-means (using the kd-tree data structure) and a method for selecting the best $K$. This method proved to be capable of finding good clusterings even for the datasets having smaller size clusters embedded in the bigger ones (which usually is a big problem for clustering algorithms).

All of the above methods deal with "hard" clustering (i.e. each point belongs to one cluster). However leaving this assumption and introducing fuzzy ("soft") clustering, where for each (point, cluster) pair a probability of the point belonging to the cluster is assigned, yields a whole set of new methods. One of the popular ones is deterministic annealing. The idea of the method is to minimize the distortion functional - $D = \sum_x \sum_y p(x,y)d(x,y)$ ($x$'s are points and $y$'s are centroids, while $d(x,y)$ is a distance between a centroid and point, $p(x,y)$ is a joined probability distribution of $x$ and $y$) subject to a constraint on the level of randomness, expressed as entropy - $H(X,Y)$.

Such a setting can be reformulated as minimization of the Lagrangian $D - TH$, where $T$ can be viewed as a temperature (by analogy with problems in statistical physics). The method of deterministic annealing (described by K. Rose in [3]) starts with high temperature values and iteratively recalculates optimal $y$ centroid locations and $p(x|y)$ - probability of associating point

$x$ and centroid $y$, until the algorithm converges (it is similar to EM) and then "cools" the system by decreasing $T$. After cooling the calculation is repeated, etc. For certain critical values of $T$ the clusters split and therefore the number of clusters increases (when $T = \infty$ all the points belong to a single cluster). Such a technique allows to avoid many poor local minima. However the issue of optimality remains.

All of the above methods deal with vector representation of a dataset. However sometimes the vector representation of the data is not easy to obtain. Then, the second setting of the clustering problem arises. One of the intuitive approaches to that problem is hierarchical clustering. It is called hierarchical because at each level a single point or group of points is joined with another, in such a way that when combined, the elements remain together throughout the remainder of the procedure. Methods of that type can be agglomerative (clumping) or divisive (splitting). All of them require distance functions for calculating distances between clusters (in order to decide which clusters to merge or to split). Agglomerative methods are usually preferred since they require less computation. Those approaches are described in [1] (Duda and Hart) in detail.

The deterministic annealing technique was also applied to pairwise clustering by Thomas Hoffman and Joachim Buchmann[4]. Another method was recently introduced by Noam Slonim and Naftali Tishby in [5] - there, the clustering of the set of points $X$ is based on the similarity of conditional distribution of another set $Z$ ($Z$ can be a set of given categories). It is shown that such a setting is equivalent to the problem of finding $Y$ - a compressed representation of $X$ , such that the mutual information between $Y$ and $Z$ is as high as possible. It yields a variational problem of maximizing $I(Y, Z)$, subject to a constraint on $I(X, Y)$. $I(X, Y)$ shows how compact the representation of $X$ is.

In [6] authors obtain an iterative algorithm for resolving the problem above. The method was applied to text categorization problems (see [7]) and showed good results. Still, the conditions of its convergence to global minimum (convexity of the sets of probability distributions) do not hold in case of clustering.

## 3 Metaclustering approach

Although most of the algorithms mentioned above enhance the quality of partitioning, none of them can be regarded as a universal solution. That is partially due to the fact that the problem of clustering is ill-defined: a particular user is always interested in "reasonable" partitioning of the data. However the meaning of the word "reasonable" depends on data representation, user's needs, and lots of other factors that cannot be easily formalized and incorporated in a mathematical model. Therefore, apart from local minima problems every clustering method will suffer from the incompleteness of a model used.

Our approach is simple: since we can not automate the choice of the optimal partitioning we will leave the final choice to user. We will present him with a small set of "candidates" to choose from.

Still, finding few "suboptimal" solutions is not an easy task. Metaclustering implies the following procedure for that: generate many different clusterings, define a measure of their dissimilarity (i.e. distance between any two clusterings) and apply some clustering algorithm to find groups of similar models. Then, the centroids of these groups will be presented as candidates for optimality and user will be able to choose from them.

The first problem we face here is generation of a large number of potentially different clusterings given the same dataset. There are many ways to achieve this goal. We can create different clusterings using different clustering algorithms. But there are not so many algorithms that we could use to obtain a big set of models to work with.

Another approach would be to pick an algorithm that could be easily manipulated. It means that the method should have some tunable parameters, such that if on some stage of the process there exist several reasonable ways to continue, changing the parameters should allow us to follow each of those paths.

Most of the algorithms described in the previous section do not possess this quality - solutions they provide are related to clear mathematical principles and changing the algorithm's behavior would make them inconsisent with the theory they are based on. $K$-mean algorithm can be manipulated by changing $K$ and starting points. However it can not be applied directly to pairwise clustering. The latter was important for our project since the protein data was available only in pairwise distance form.

Another reason not to use $K$-mean is that the optimal clustering (found by $X$-Mean algorithm or with Bradley and Fayyad method) will probably be the best one among the ones $K$-mean can find. But what if the imperfection of the output clustering is mostly due to the poor data representation? Then it might happen that no local minima found by any of the $K$-mean algorithms will satisfy a user.

Since one of our major concerns was flexibility, we have chosen the modified version of the hierarchical agglomerative clustering to work with.

Hierarchical agglomerative clustering (HAC) is a sequential algorithm based on pairwise distances between points in the given dataset. It starts with the assumption that each point represents a single cluster, so at the beginning the number of clusters is equal to the number of points in the dataset. The algorithm proceeds by merging the two clusters that have the smallest mean distance between each point in one and each point in the other (actually there exist many criteria for merging, but this one seems to be the most accurate (see [1] for details)). The clustering is charachterised by a dendrogram - a clustering tree, showing how the samples are grouped.

The algorithm has a property that if two points are merged into one cluster at some moment, they will remain in the same cluster for the rest of the process. The way we make this deterministic algorithm produce different results is "bagging": on each step of the HAC algorithm (i.e. merging of two clusters) we choose a subset of the whole set of existent clusters and chose a pair from this subset to merge. Different algorithms that select this subset are descibed in detail in the next section. What is important now is that although the bagging technique allows us to obtain different results, they should remain "reasonable". The last statement can be illustrated by the following example: suppose we have a pairwise representation of the dataset, such that not all features are reflected by the set of distances and there exist several qualitatively different ways to cluster the data points. Choosing a relatively large subset of points will preserve most of the information of the original set. Therefore the merge that will be performed cannot be completely irrelevant. Since the new subset is chosen on every step, the clustering will not be biased, and since different experiments will use different subsets, we will be able to obtain many models.

Another method to generate different clusterings is based on metric changes. In order to do that we need to perform the multidimensional scaling on the set of pairwise distances. It allows us to find the principal components of the dataset and map a set of points to a set of vectors in a low-dimensional space, preserving more or less the distances between points (the protein data was found to have 6 principal components).

The hope is that the principal components correspond to the main features that are responsible for the current set of distances. Then (if it is true) reweighting the principal components (e.g. stretching or shrinking the dimensions of the vector space) recalculating the distances set and clustering would give us an ability to increase or decrease the relative importance of each of those factors and thus will probably allow us to overcome the imperfection of the model. Both approaches were applied to the protein data and synthetic 2-dimensional dataset data, generated from mixture of normal distributions (the latter was done to visualize the results of the work of our algorithm). Additional discussion on those methods is provided in section 4.

### 3.1 Family of Bagging Algorithms

We now present a family of "bagging" algorithms that are capable of generating different clusters given the same initial data set.

First and the simplest algorithm is "random bagging". It starts by randomly selecting a subset of points, i.e. a "bag" of points. Then, the points in the bag are forced to merge until a certain number of clusters is reached. After the predefined number of merges have occured, the rest of the points that were not in the bag are allowed to join creating a clustering.If the random number generator is adequate it is clear that several different initial sets of points may be created. Different sets of initial points will, potentially, capture different aspects of the data presented to the algorithm and may lead to different final clusterings of the same data.

Note that the number of points to be put in the bag and the number of merges forced before other points are allowed to join have to be arbitrarily selected by the user. It is important to see that we are generating different clusterings by creating artificial situations by selecting random subsets of points to be merged first. The points and clusters in the bag do not have information about all clusters/points in the dataset, only about the ones that were selected into the bag, i.e. clusters/points in the bag are forced to be merged with the **closest available** clusters. It is clear that if the number of points in the bag is very low, then there is less chance to create potentially different clusterings, since the changes will occur only on the detail - noise level when there are still a lot of clusters/points to be merged, so we may not see as much difference on the top levels. Also, if we select too few merges to be performed in the bag, then there is a good chance that the number of points that we were forced to merge against their will is small and the rest of the merges will happen "naturally", i.e. according to the same strategy as the original greedy hierarchical clustering algorithm, at the end producing similar results. It is more interesting to look at the clusterings generated by selecting more than 50% of points into the bag and by forcing more than 50% of merges in the bag before other points are allowed to join. If we were to force too many merges, then when allowed the points that were not in the bag will be merging either between themselves or with one huge cluster, which may result in less meaningful clustering.

We have also implemented a so-called "hierarchical bagging". The algorithm differs from the previously described by the initial bag selection method. In this new algorithm, the first step is to create the original clustering using all data by greedy hierarchical clustering algorithm. The next step is to pick a desired number of points close to centroids of clusters from different levels in the original clustering tree. The number of points in the bag should be selected following the same logic as described for the random bagging algorithm. The reason behind this approach was to capture variations in the structure of the original model, i.e. since we are allowing relatively distant points to merge we might generate a lot of large clusters, then the rest of the points will merge to form smaller clusters.

The experiments have shown that both of the methods described above have one major disadvantage. They concentrate on making changes at the "bottom level" hoping that the effect would trickle up. However, since there is no influence on merges at the higher levels, i.e. we allow the greedy algorithm to merge the closest clusters first and then merge the more distant ones, the changes at the higher levels will be minimal or none, depending on the dataset.

The solution to this problem is the third and the final algorithm from the "bagging" family. The new approach is to "bag and merge" at all levels. At each level, we select an arbitrary number of clusters to be put in the bag, for e.g. we choose to randomly select 50% of existing clusters, it will later be shown that the performance does not depend on the number of clusters in the bag, and perform one merge according to the rules of greedy algorithms (merge the closest clusters). The procedure stops when all points were clustered or the desired number of clusters was achieved. The randomness in selection of points into the bag assures generation of different clusters on higher levels, i.e. the algorithm guarantees to produce many different clusterings, which is our goal.

# 4 Meta Clustering

As stated above, the main goal of this work is to generate a set of qualitatively different clusterings of the same data and present them to the expert. Therefore, a technique to determine how different the clusterings are is required.

To estimate the dissimilarity of different clusterings we define a function $f : C \times C \mapsto \mathbb{R}$, where $C$ is a space of clustering models (i.e. our hypothesis space). It allows us to abstract from the complicated nature of the objects we are dealing with and work with a set of $f$ values (i.e. distances). Then, having a set of distances between different models we can apply one of the clustering algorithms to obtain a compact representation of the clusterings' set (few qualitatively different models).

Several different types of $f$ function are tested in this project. We are somewhat limited in our choice due to the fact that the protein dataset is not available in vector form (we could use the results of multidimensional scaling for the protein dataset to overcome this problem in our particular case, but that would make our methods dataset dependent). Therefore, we cannot use such popular measures as KL-divergence. However several other metrics can be used for that purpose.

## 4.1 Hop Distance

The hop distance is based on the dendrogram (clustering tree) of hierarchical agglomerative clustering algorithm. Let us first explain the notion of dendrogram: All algorithms derived from the hierarchical agglomerative clustering generate a sequence of merges of data points. This sequence of merges may be represented as a binary tree. Each leaf node of this tree is a singleton cluster containing only one point and each merge performed is represented as a non-leaf node connecting two clusters that were merged.

The set of points contained in a cluster is the set of descendants of the cluster node.

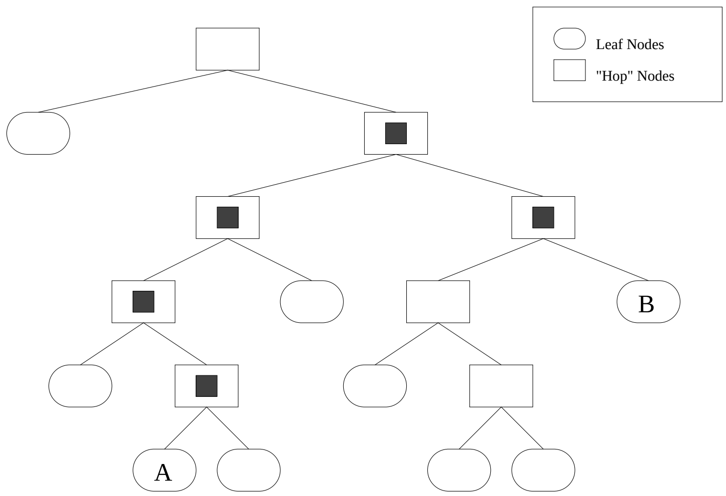


Figure 1: Representation of Hop Distance metric using Dendrogram

The hop distance is precisely a distance between points in this binary tree - the number of hops required to reach one point from the other. An analogy to routers in a network may be used to visualize it. If two leaf nodes are two network terminals and each non leaf node is a router the smallest number of hops, or routing decisions, required to send a package from one terminal node to the other is the hop distance among the two nodes. As an example, in the figure presented above, the hop distance among data points A and B would be five.

The hop distance can be used to compare different hierarchical agglomerative clusterings: it is intuitively understandable that if in one clustering model two points are merged very fast and in another one they remain in different clusters till the last merge, those clusterings are likely to be different.

Suppose now that we want to compare 2 different hierarchical clusterings of the same dataset. First consider hop distances between all pairs of points in a dataset in both dendrograms, corresponding to these clusterings - $H^1_{ij}$ and $H^2_{ij}$.

For any pair of points $i$ and $j$ $H^1_{ij} - H^2_{ij}$ shows the difference in the lengths of sequences of merges that make $i$ and $j$ one cluster. Calculating the sum of those differences over all $i$ and $j$ will yield a measure of dissimilarity of dendrograms (i.e. dissimilarity of clusterings). It also makes sense to sum the squared differences to penalize the bigger differences between points even more. Therefore the measure of similarity of clustering models can be defined as

$$H(C1, C2) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (H^1_{ij} - H^2_{ij})^2$$

The $\frac{1}{n^2}$ is introduced in order to make the measure independent of the size of the dataset.

Another important idea is that when comparing two clusterings we should not calculate the hop distance from the leaves of the dendrogram (bottom of the tree), but instead from the level the particular clustering has reached. We will illustrate it by the following example: suppose we are comparing 2 clusterings of the same data, obtained with a HAC algorithm. Suppose both clusterings have the dataset partitioned into five groups (clusters). If the groups are identical the dissimilarity of the clusterings should be 0. However, this will not the case with the given hop distance measure when those groups are obtained through different sequence of merges. Redefining $H_{ij}$ so that only the merges between currently existing clusters are considered will solve this problem: if the points $i$ and $j$ are in the same cluster $H_{ij}$ will be 0 if they are in the clusters that will be merged on the next step it is equal to 2, etc. This redifinition actually means that when comparing clusterings we are not interested in how we have obtained them, but how they are different between themselves.

The Hop Distance has a set of nice properties that justify its application to the problem at hand:

1. It is symmetrical: $H(C1, C2) = H(C2, C1)$
2. $H(C1, C1) = 0$
3. The triangle inequality holds (the above properties mean the hop distance is a metric).
4. Changing the dendrogram on upper levels changes the Hop Distance more than changing on lower-levels.

All of those properties can be easily verified.

### 4.2 Cluster difference

Another measure of dissimilarity of clustering models can be obtained with the following simple calculation: define $d_{ij}$ as 0, if two points are in the same cluster in both clusterings and 1 if they are in different clusters. $\sum_{i,j} d_{ij}$ would be the disimilarity of 2 clustering models.

Such a measure is a metric as well (we will call it Cluster difference). An important property of that metric can be illustrated by the following example: Suppose that the difference between two clustering models is that one of the clusters of one model is split among 2 clusters in another model. And in the third model it is just "torn apart" by many clusters. The first two models have smaller Cluster difference than any of the two and third (and that is natural). It should be noted that the hop distance metric has the same property.

### 4.3 Mean internal distance

The last measure of clusterings' dissimilarity is the mean internal distance (MID). MID for a given cluster is defined as an average distance between all pairs of its points. In fact it measures how "happy" the points are (i.e. how homogeneous the group they are merged to is). Also average MID for the whole clustering model can be calculated (as an average of weighted MID's of individual clusters of the model).

With a hierarchical agglomerative clustering algorithm we can plot a graph showing the change in the model's MID from 0 (when all points belong to different singleton clusters) to the average distance between points in a dataset (when all points merge to one cluster), as merges occur. See figure 2 for example of such a plot. The "jumps" on the graph show the natural number of clusters for the dataset - when a big jump occurs it means that the last step merged two really different and separated clusters (and since the ones to merge are the closest ones others are even more separated).

MID vs number of clusters graph characterizes the current model and it can also be used to compare clustering models. One of the possible choices for dissimilarity measure is the integral of MID difference over a range of merges (i.e. $\sum_{i=n}^{1}(MID_1(i) - MID_2(i)($, where n is the number of points in the dataset and $MID(i)$ is the model mean internal distance on $i$-step).

### 4.4 Comparing different metrics

Of the above 3 metrics the Hop distance is definitely the most discriminative (it makes it easier to compare clustering models), while MID can hardly be considered a good measure, since very different (from human point of view) clusterings can have similar MID plots. The Cluster difference metric and Hop distance are similar in the way the estimate the dissimilarity of the models. Hop Distance is potentially better, since it captures more information about the ways different clusters will merge and thus can distinguish between models that may be considered similar by Cluster difference metric.

We have tested both metrics and found that they yield similar results.

### 4.5 Considerations Regarding Meta-Clustering

We have used the same agglomerative clustering algorithm for metaclustering. We could have used any other method instead. However we assume that any reasonable algorithm should work fine - since the process of generating of different models was guided by gradual change of model parameters we may hop that the patterns in the metaclustering data should be relatively easy to discover.

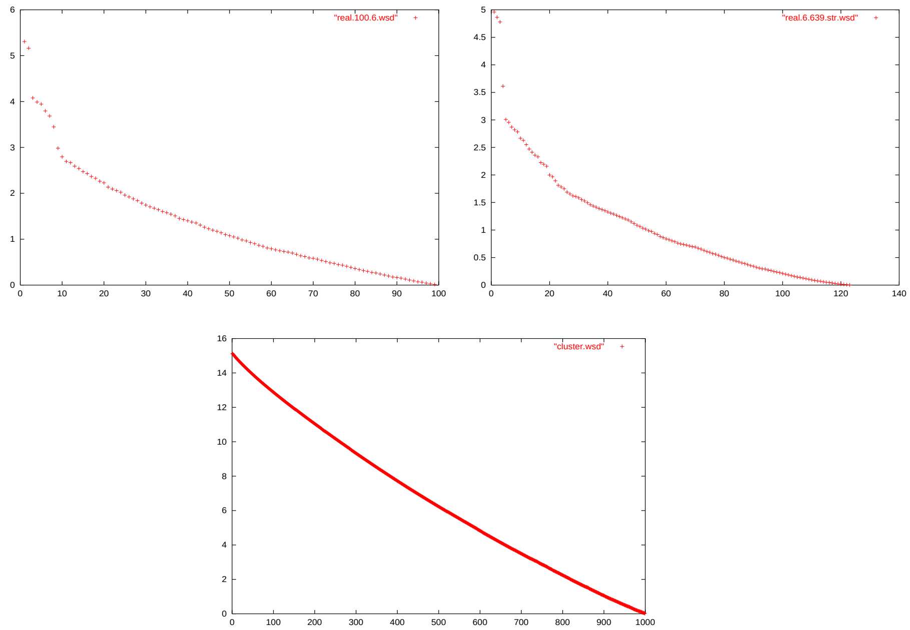


Figure 2. MID for meta-clustering of protein clustering models obtained using “bag and merge”, metric changing and random merges

The fig. 2 shows the MID plots for metaclustering obtained with bag-and-merge and metric changing methods. Big "jumps" occuring when the number of clusters becomes low show that we really have several groups of potentially different clusterings. For comparison we have included the same plot in the case where clustering models were selected randomly (any possible clustering models, performing random merges, were allowed) - the smoothness of the MID curve ensures that the hypothesis space is uniform.

We explain the existence of expressed clusters of models below: Consider the metric changing method first: it is obvious that small changes in the "stretching" parameters can change the order of top-level merges and thus result in a very different clustering. In a way the distance function (Hop distance or any other) is not continuous on the tuned parameters. The same occurs with bagging. That is why we can hope to find very different clustering models (in the next section we show the experimental results, confirming it).

Since we use HAC algorithm for metaclustering, it’ s the natural representation is a dendrogram (Fig.3):

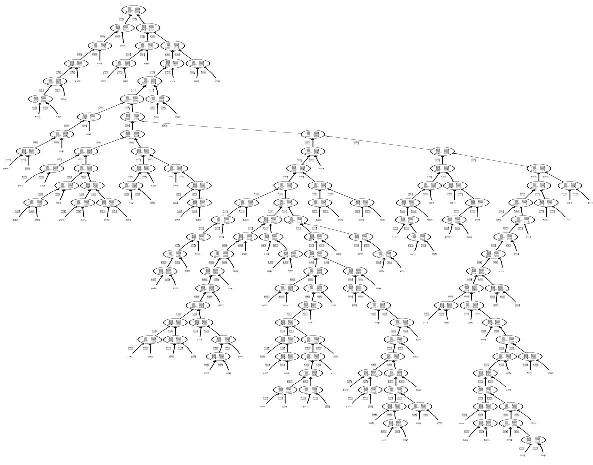

Figure 3: Meta-clustering Dendrogram for Protein Data

Although we could also apply multidimensional scaling to distances between clustering models and plot the projection into 2 principal dimensions, the picture would not be very clear: unlike many real datasets the metaclustering one is really high-dimensional and it will look as a very dense cluster with few outliers in any 2-dimensional projection.

## 5 Experiments

The bagging algorithms were tested on real and synthetic datasets, where the synthetic dataset is a mixture of six gaussians in two dimensional space as shown below. In order to ensure that our method finds really different but reasonable clustering models we have conducted the following experiment: we applied various bagging algorithms to obtain a partitioning of the dataset into several parts. Then with a metaclustering algorithm we found the centroids of clusters of models (the "optimal" number of clusters was chosen with the help of MID plots). Few of those selected clusterings are presented below (we have included those, containing three clusters, however we have similar results for larger numbers of clusters as well). Of course most of those clusterings are not "optimal" by any criteria, but they can be considered reasonable and presented as candidates to user. It was also found that Hop distance and Cluster difference metrics yield similar models. Results, obtained with Hop Distance and Cluster difference metrics are plotted on Fig.4 and Fig.5 respectively(see Appendix 1).

The same experiments were repeated with the protein dataset. The few clustering models, selected by "bag-and-merge" algorithm, are plotted on Fig 6 (see Appendix 1). They pictures are not so clear, since the data is truly 6-dimensional, while we are plotting only 2 dimensions. Figure 7 present the clusterings obtained with with metric changing algorithm.

# 6 Conclusions and future work

We have presented a new method for finding qualitatively different clustering models of the same dataset. That was confirmed by experiments both with synthetic and real data. Though it was the first such approach to meta-clustering the results are encouraging. We hope that more work will follow.

There are possibilities for further research in this direction. First of all, algorithms other than hierarchical agglomerative clustering might be used. Also, one could utilize other metrics for comparing models. An important issue that may also be persued is computational complexity. The described hierarchical algorithm is relatively slow, i.e. in its fastest implementation it is of order $n^2 \ln n$. The proposed distance measures for clustering models are based on point-to-point comparison and require a lot of computation. Changing the method and distance measure may allow to overcome this difficulty.

# 7 References

[1] Duda R. Hart P. (1973). Pattern classification and scene analysis. John Wiley & Sons.
[2] Pelleg D. Moore A. (2000). X-means: Extending K-means with Efficient Estimation of the Number of Clusters
[3] Rose K. (1998) Determenistic annealing for Clustering, Compresion, Classification, Regression and related optimization problems. Proceedings of IEEE,86(11), 1998
[4] Hoffmann T. Buchmann (1997) Pairwise data clustering by deterministic annealing. IEEE Transactions of PAMI, 19(1):1-14, 97
[5] Slonim N. Tishby N. (1999) Agglomerative Information Bottleneck. In Proc. of Neural Information Processing Systems
[6] Pereira F. Bialek W. Tishby N. (1999) The information bottleneck method. In Proc. of the 37-th Allerton Conference on communication and Computation

[7] Slonim N. Tishby N. (1999) Document clustering using word clusters via the information bottleneck method.
[8]Bradley P. S. Fayyad U.M. (1998) Refining initial points for K-Means clustering. Proceedings of the 15-th International Conference on Machine Learning.

# 8 Appendix 1

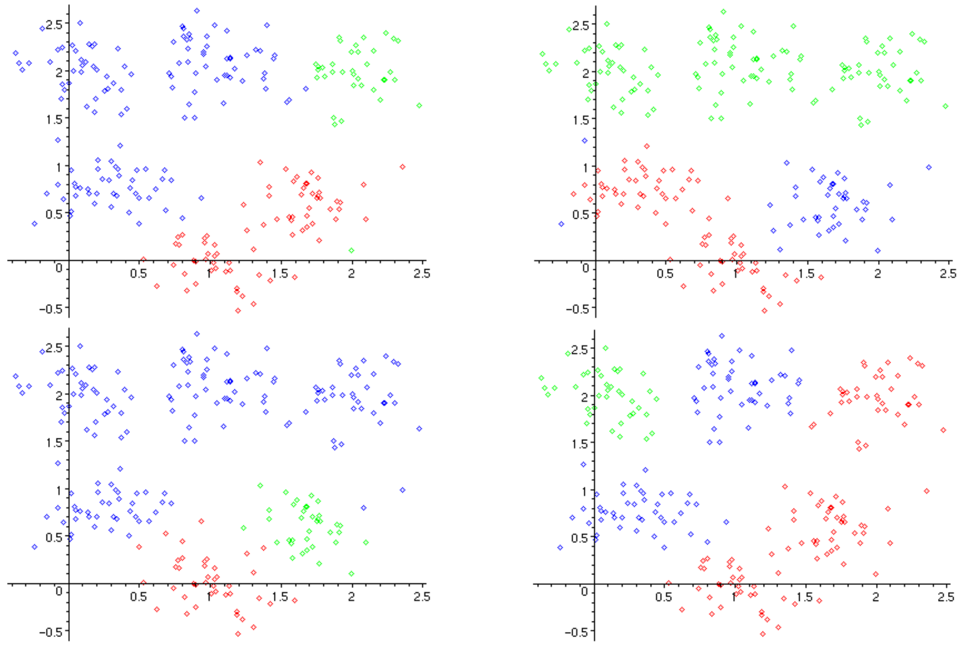

Figure 4: Clusterings of the synthetic data obtained using “bag and merge” algorithm and Hop Dist metric

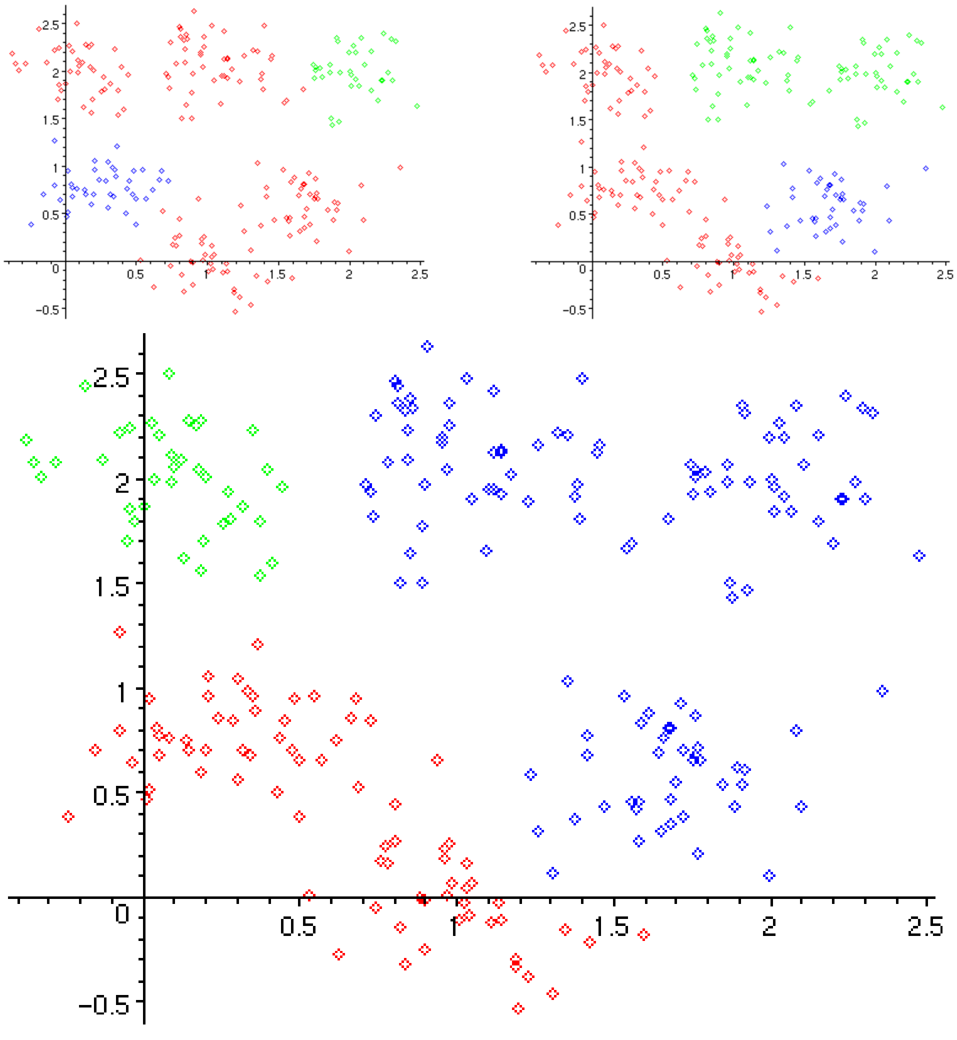


Figure 5: Clusterings of the synthetic data obtained using "bag and merge" algorithm and Cluster difference metric

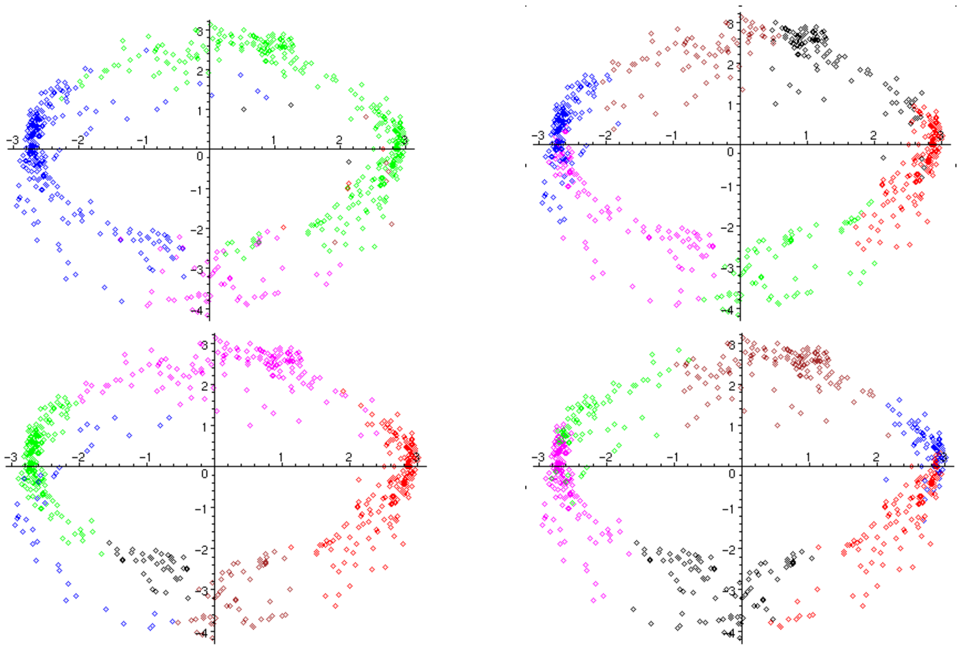

Figure 6: Clusterings of the protein data obtained using metric changing

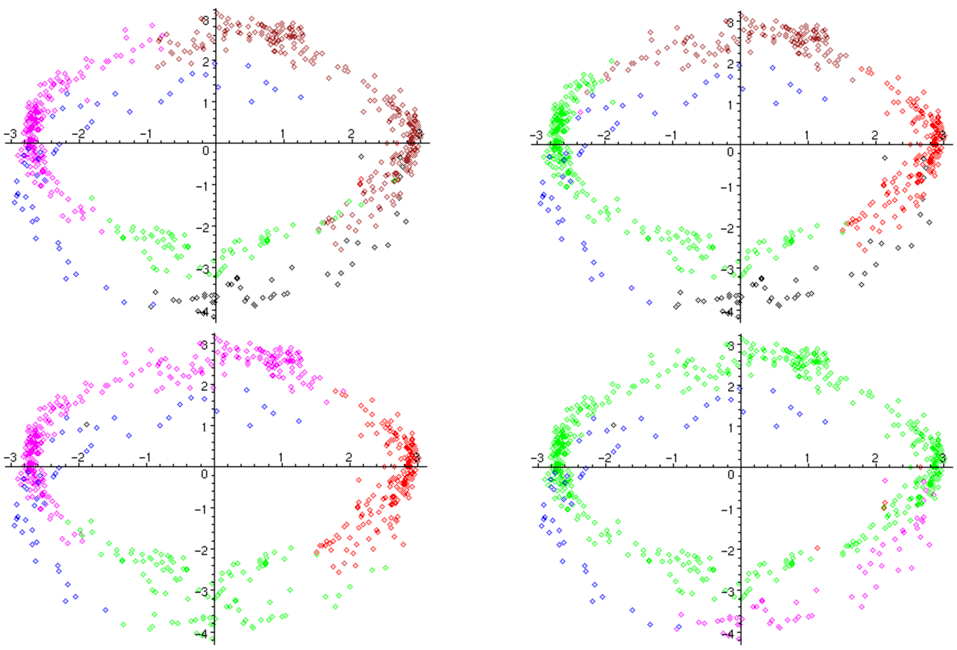

Figure 7: Clusterings of the protein data obtained using “bag and merge” algorithm

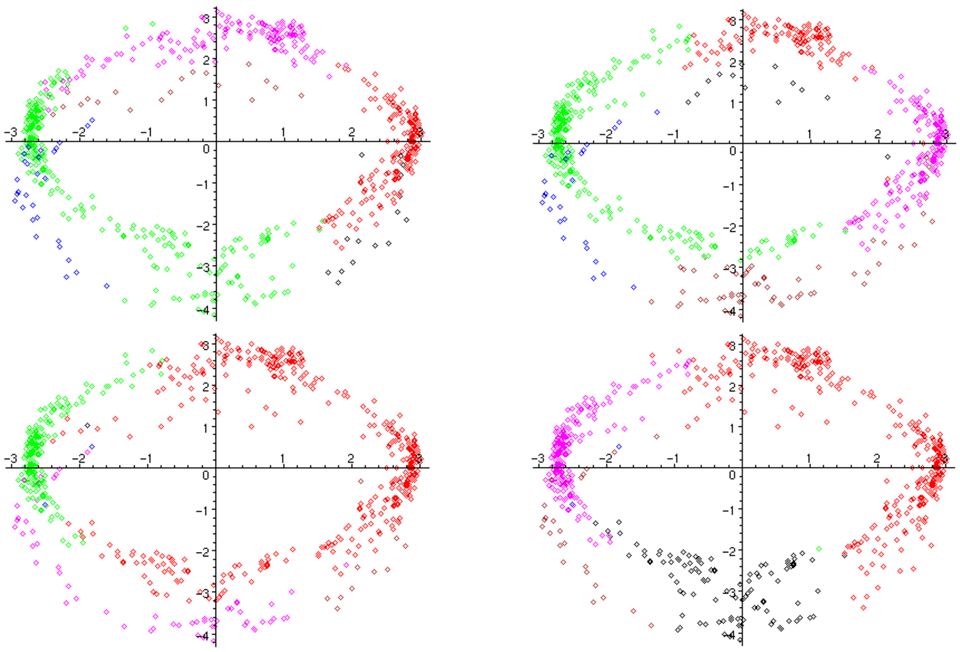

Figure 8: Clusterings of the protein data obtained using "bag and merge" algorithm : 2