On the Effects of Constraints in Semi-supervised Hierarchical Clustering

Hans A. Kestler[1,2], Johann M. Kraus[1,2],
Günther Palm[1], and Friedhelm Schwenker[1]

[1] Department of Neural Information Processing, University of Ulm,
89069 Ulm, Germany
hans.kestler@uni-ulm.de, friedhelm.schwenker@uni-ulm.de
[2] Department of Internal Medicine I, University Hospital Ulm, Robert-Koch-Str. 8,
89081 Ulm, Germany

Abstract. We explore the use of constraints with divisive hierarchical clustering. We mention some considerations on the effects of the inclusion of constraints into the hierarchical clustering process. Furthermore, we introduce an implementation of a semi-supervised divisive hierarchical clustering algorithm and show the influence of including constraints into the divisive hierarchical clustering process. In this task our main interest lies in building stable dendrograms when clustering with different subsets of data.

1 Background

The aim of cluster analysis is to explore a collection of data items and to group *similar* objects together. Similarity can be measured in many different ways, for example by the Euclidean distance. Every information used in an unsupervised learning technique comes from the data themselve, which means no external teaching signal guides the algorithm. During the past years several modifications were proposed to incorporate additional background knowledge into clustering algorithms [1–9]. The main idea is that in some tasks prior information about a small amount of data is known and should support the clustering process. This background information is usually provided as a set of constraints between pairs of data items [2]. Real class labels seem not feasible in clustering because of the inherent discrepancy between labels and clusters. Pairwise constraints are typically encoded as *must-links* and *cannot-links* [2] either indicating two points belong to the same cluster or to different clusters.

Up to now different clustering algorithms were adapted to make use of semi-supervised clustering. Constraint-based approaches modify the cluster search, either by including constraints into the objective function [1] or by initalizing and constraining during the clustering process [10, 3, 9]. Metric-based approaches first train the cluster metric to satisfy the constraints using shortest-path algorithm [4], expectation maximization [5], gradient descent [6], convex optimization [7, 8] or combinations of constraint-based and metric-based methods [11].

F. Schwenker and S. Marinai (Eds.): ANNPR 2006, LNAI 4087, pp. 57–66, 2006.

Many real-world unsupervised learning tasks may profit from an inclusion of background knowledge, e.g. image processing, text mining or even lane finding using GPS data. Another important field for data mining using semi-supervised clustering is the functional grouping of genes. For this problem we are not only interested in one particular clustering as built by partitioning clustering algorithms, such as K-means or SOM, but in getting a hierarchy of partitions (dendrogram).

Our main emphasis lies on building robust hierarchical clusterings for DNA microarray data. Here we are interested in the resulting dendrogram and focus on the question of creating stable dendrograms. In this setting one of the problems that frequently arise is the change of the branching structure when computing several runs with different data subsets. Since the inclusion of background knowledge seems to enhance the accuracy of clustering algorithms, we studied its effect on the dendrogram stability. In Section 2 we explain some basic annotations on the effects of constraints in semi-supervised hierarchical clustering. In Section 3 we introduce our new algorithm for divisive hierarchical clustering with a-priori information and section 4 shows our results from semi-supervised clustering in hierarchical algorithms.

2 Background Knowledge in Hierarchical Clustering

2.1 Constraints

Background knowledge may be available in different forms, e.g. as labels or constraints. Following Wagstaff & Cardie [2] we use *cannot-link* and *must-link* constraints to insert a-priori information into our clustering algorithm. We don't use labels, because they indicate class memberships and a class can consist of different and distant clusters. As mentioned above, *must-links* indicate two data items being arranged in one cluster and *cannot-links* allude not to assign two data items to the same group.

2.2 Effects from Constraints in Hierarchical Clustering

Constraints indicate a relationship between two data items. When a partition of a data set into k cluster is computed, we may profit from including this form of background knowledge, because constraints seem to support the clustering algorithm to define the clusters of this partition more accurate. Computing a hierarchy of partitions is a different task. Here we are confronted with a sequence of refinements of clusters. A hierarchy of partitions is either computed by a subsequent assembly of two clusters or by a subsequent division of one cluster. Whereas each partition is formed by a different number of clusters there is a different set of relations between the data items in these clusters, i.e. every partition provides its own set of constraints, see Figure 1. In the following we present two scenarios which should point up the effects of constraints in hierarchical clustering.

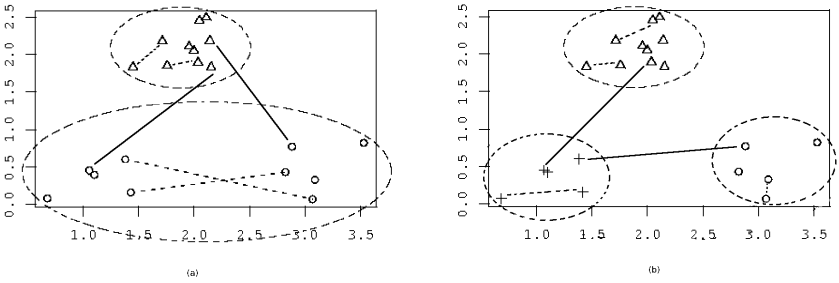


Fig. 1. Basic example for a data set with two clusters (Subfigure a, clusters △, *o*) and its refinement into three clusters (Subfigure b, clusters △, + , *o*). Additionally each partitioning provides its own set of constraints (lines = *cannot-links*, dashed lines = *must-links*).

Effects from *Cannot-Links*: In our first scenario we want to show the effects of *cannot-links* in hierarchical clustering. Imagine we want to compute a dendrogram for a data set using a divisive hierarchical clustering algorithm. To simplify matters we only look on the first two levels of the computed dendrogram. These two levels propose two partitions of the data set. On the first level we see a partition into two clusters and on the second level one of these clusters is split again. Additionally we can provide two different sets of *cannot-link* constraints.

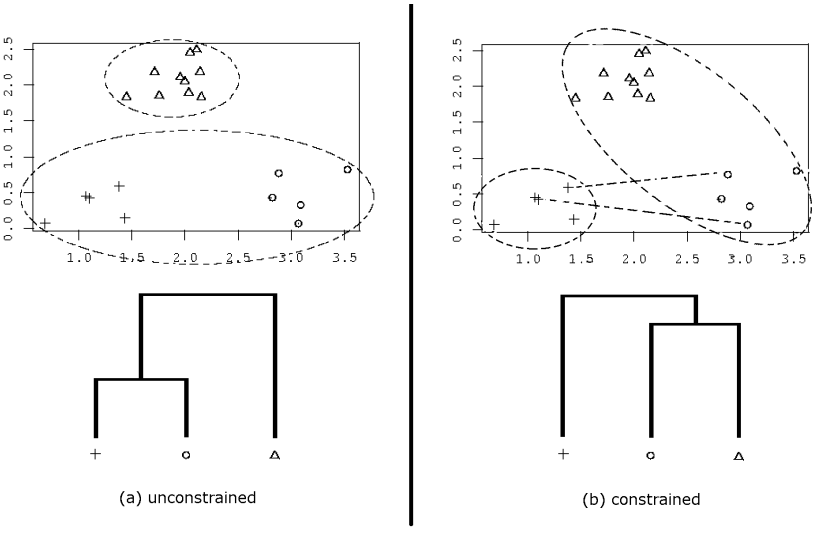


Fig. 2. Different branching structures when providing a few *cannot-links* between clusters on a lower level. Subfigure (a) shows a clustering into two (dashed circles) and three clusters (△, + , *o*) and the corresponding dendrogram. Adding *cannot-links* (dashed lines) between some elements from cluster + and cluster *o* results in a different clustering on the second level, see Subfigure (b).

The first set indicates relationships between data items from the partitioning into two clusters and the second set contains *cannot-links* from the partitioning into three clusters.

When incorporating the first set of constraints we may get a more accurate clustering into two clusters, because *cannot-links* for data items on the boundaries of the clusters could guide the algorithm to improve the shape of the clusters. Since all of these *cannot-links* could be resolved in the first clustering step, we don't use them in the further steps any more.

On the other hand we probably want to make use of the second set of constraints. The possible effect of the inclusion of this set of constraints could be seen in Figure 2. In this example we see two initial clusters, one of them builds a supercluster that could be refined again. After providing some *cannot-links* indicating this refinement in the first step of the clustering procedure the algorithm tries to resolve even these *cannot-links*. As a result either the constrained items could not be assigned to the correct supercluster or actually a complete different supercluster is built.

Effects from *Must-Links*: In this scenario we want to describe the effect of adding *must-links*. Again we only look at the first two levels of the dendrogram, but this time we only have one set of constraints containing *must-links* deduced from the first level. Similar to the inclusion of *cannot-links* we can distinguish between two effects. When we use *must-links* from clusters on the second level, the dendrogram on the first level may profit from a finer definition of the shapes of these two clusters.

Contrary to the above, in the next refinement steps these constraints hinder the clustering algorithm. Figure 3 shows the results from adding *must-links* to the divisive clustering algorithm. In this example two *must-links* could not be correctly split.

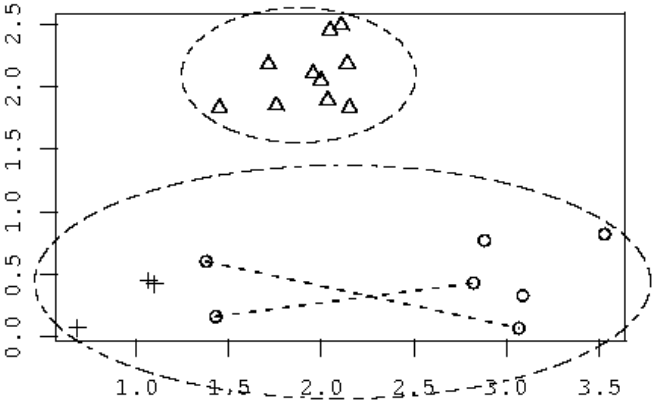


Fig. 3. Partitioning into three clusters ($\triangle$, $+$, o) after including *must-links* (dashed lines) between elements from a partitioning into two clusters (dashed circles). When splitting the lower cluster the two *must-links* hinder a correct refinement.

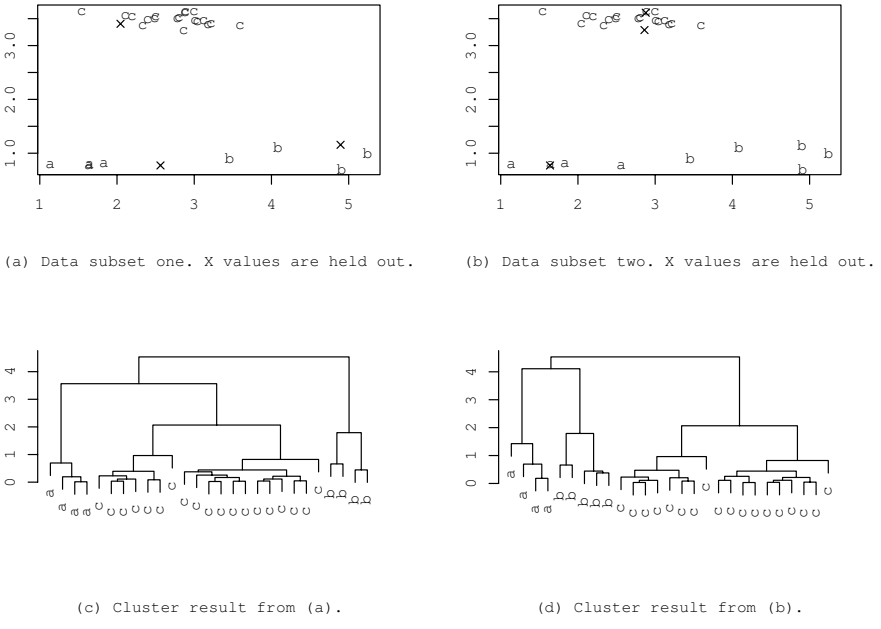
3 Semi-supervised Divisive Hierarchical Clustering

In microarray data mining studies we are usually confronted with low-cardinality data. Computing dendrograms after removing or adding some samples could easily contradict a previous constructed hierarchy.

Figure 4 shows a simple two-dimensional example for this behaviour. Three clusters were formed using Gaussian distribution. Randomly removing three items results in different subsets of the data. Subfigures (a) and (b) show two possible subsets. A hierarchical clustering of these two subsets results in two contradictory dendrograms, where either cluster **a** and cluster **b** or cluster **a** and cluster **c** are proposed to be similar. Applied to the task of gene functional grouping, no unique branching structure can be predicted when working on incomplete data sets, even though a correct clustering is found. In this section we introduce our attempts to enhance the dendrogram stability by including background knowledge into the clustering process.

3.1 Constraints

According to the considerations described in Section 2 we decided to restrict the use of constraints. To avoid mixing constraints from different levels of the



(a) Data subset one. X values are held out.

(b) Data subset two. X values are held out.

(c) Cluster result from (a).

(d) Cluster result from (b).

Fig. 4. Subfigures (a) and (b) show two different subsets of a simple two-dimensional data set containing clusters **a**, **b** and **c**. The subsets are built by holding out three items, for clarity marked by X. Clustering both subsets results in two different dendrograms. In Subfigure (c) cluster **a** and **c**, in Subfigure (d) cluster **a** and **b** are deduced from one combined cluster.

dendrogram we only add constraints for one particular partition of the data set. Furthermore, we use our set of constraints only for the clustering on this predefined level. This means for example providing a set of constraints that include *cannot-links* for a partition of three clusters, we only use these constraints on the second level of the dendrogram. In the task of building more stable dendrograms, we are mainly interested in the dendrogram stability on the first level. Therefore, we additionally decided to use only constraints from the first level. Nevertheless, our algorithm could easily be expanded to make use of constraints from different levels of the dendrogram. Furthermore, we build the combined transitive closure [10] over all constraints to get the most from our background knowledge and to avoid presenting contradictory constraints.

3.2 Algorithm: Constrained-Divisive

Divisive hierarchical clustering starts the clustering procedure with the complete data set in one cluster. The largest available cluster is then at each subsequent step divided into two clusters until finally all clusters contain only one single data item. To identify the largest available cluster, usually the diameters of all clusters are compared. To find out where to split the chosen cluster, its element with the largest average distance to all other elements can be used as the initial item of the new cluster. According to a user-defined distance measure all remaining points are assigned to their most similar cluster.

Our implementation of CONSTRAINED-DIVISIVE modifies this basic divisive algorithm. Only on the first partitioning, i.e. the clustering into two clusters, we have to include the constraints, on all other steps the constraints are not used

Algorithm 1. Constrained-Divisive

CONSTRAINED-DIVISIVE(data set $D = \{d_1, \ldots, d_n\}$, *must-links* $Con_{=} \subseteq D \times D$, *cannot-links* $Con_{\neq} \subseteq D \times D$)

1. Let C be the initial cluster group: $C = \{D\}$.
2. Select the cluster $C_m \in C$ with the largest dissimilarity between any two of its objects. Divide C_m following (3) to (7).
3. Select the element s_z with the highest average dissimilarity to all other elements and, if clustering the first level, with an unresolved *cannot-link*. s_z initiates the splinter group S. If clustering the first level:
   $\forall d \in C_m, s \in S$: If $(d_i, s_j) \in Con_{=}$, move d_i to S.
4. $\forall d$ still $\in C_m, s \in S$: Compute the difference of distances:
   $Diff(i) = [averagedistance(d_i, d_j)] - [averagedistance(d_i, s_j)]$.
   Select the element d_h with the greatest difference $Diff(h)$.
   If clustering the first level: If $\exists s_j \in S : (d_h, s_j) \in Con_{\neq}$, set $Diff(h) = 0$.
   If $Diff(h) > 0$, move d_h to the splinter group S.
5. Repeat (4) and (5) until all differences $Diff(h)$ are negative.
6. Now the original cluster ist split into two clusters. One is the splinter group S, the other is formed by the remaining elements in C_m.
7. Iterate between (2) and (7) until all clusters $\{C_1 \ldots C_k\}$ contain only a single element.

any more and we compute the dendrogram according to the basic algorithm. In the following we comment the clustering process from the first step more precisely. We include *cannot-links* into two decisions. When looking for the first element to split we could use an element with a *cannot-link* and a large average distance to all other elements. Secondly we avoid violating *cannot-links* when assigning elements to a chosen cluster. In addition we use *must-links* when assigning elements to a new cluster by moving all linked elements at once. This proceeding guaranties not not violate any constraints during the clustering into two clusters.

Our clustering algorithm could easily be expanded to make use of constraints from different levels. Therefore, one set of constraints for each level and an indicator for the context (the level) of this set must be included. Every set of constraints then should only be used in its context.

4 Experiments and Results

4.1 Data Sets

The first data set we analyzed is an artificial example of a two-dimensional problem containing three different clusters. We built each cluster by the use of a Gaussian distribution as seen in Figure 4. A partition into three clusters is easy to compute, because the clusters are obviously separable. Nevertheless, when removing some data items the partition into two clusters is not stable.

Our second data set comes from gene expression profiles from the pancreas (see Buchholz et al. [12]). This data set provides two classes (pancreas cancer vs. pancreatitis/normal) for 62 samples. Each sample is characterized by 169 gene expression values.

4.2 Evaluation Method

There is no gold standard for evaluating clustering results and therefore it is necessary to define the basic problem. In our study we want to compare the results from different clusterings. Jain & Dubes [13] suggest to use a relative criterion to measure the agreement between two clusterings. We do not measure the totally agreement of two dendrograms, but the agreement on a selected partition. As we have the a-priori information about the cluster membership of all samples on the first levels, we can measure the stability of the clustering algorithm by comparing the cluster results with this *external* clustering. On the other hand we can measure the stability on the lower levels by comparing all clusterings from the different subsets with each other.

We make use of the Rand Index [14]. Given a set of n items and two partitions P and Q there are four ways to compare clusters in these different partitions. Let a be the number of all pairs of items in the same cluster in P and in the same cluster in Q, b be the number of all pairs of items in the same cluster in P but not in the same cluster in Q, c be the number of all pairs of items not in the same cluster in P but in the same cluster in Q and d the number of all pairs of items in different clusters in both partitions. The scores a and d therefore

represent agreements, the scores b and c disagreements in the partitions. The Rand Index is the sum of the correct decisions compared to all decisions:

$$\frac{a+d}{a+b+c+d} \tag{1}$$

The value for perfect agreement of two partitions is 1. One advantage of computing the Rand Index is the possibility to include the constraints into the evaluation result. As we count pairwise clustering decisions, we omit the constrained pairs from the computation of the Rand Index. This Corrected Rand Index is a better indicator for the improvement of the clustering algorithm, because it only counts *real* decisions.

4.3 Results

For our artificial data set we only measure the stability on the first level. Here we want to get a more stable partition into two clusters. We compare all results from CONSTRAINED-DIVISIVE to the external label. Table 1 shows the results for this test. After removing randomly 10% of the data items ten times and adding 3% and 5% random constraints we computed the Corrected Rand Index for each run. By adding constraints we could improve the stability of the dendrogram on the first level.

Table 2 shows the results for the microarray data set. Here we not only measured the stability on the first level of the dendrograms but also analyzed the effect from the constraints on the stability on the lower levels of the dendrograms. We computed the Rand Index for the results on each level over all runs.

Table 1. Stability results for CONSTRAINED-DIVISIVE on the first data set. After removing 10% of the data items and adding 3% and 5% randomly constrained pairs from partitioning into two clusters the stability on the first level increases. The value of the Corrected Rand Index is the median from 50 runs. The second value counts the number of correct partitionings into two clusters according to the external label.

Constraints	Corrected Rand Index	Correct decisions
without	0.89	23
3%	1	28
5%	1	33

Table 2. Stability results for CONSTRAINED-DIVISIVE on the pankreas data set. After removing 10% of the data items and adding 5% and 10% constrained pairs from partitioning into two clusters the stability on the first level increases. The value of the Rand Index is the median from 50 runs. The further rows show the median for the Rand Index from comparing the different partitionings on the lower levels.

Constraints	Rand Index	Level 5	Level 10
without	0.96	0.96	0.94
5%	1	0.98	0.93
10%	1	0.98	0.94

The median from these values indicates that the constraints improve the stability even for some lower levels.

5 Discussion and Conclusion

Clustering of gene expression profiles is an important task in DNA microarray analysis. As hierarchical clustering algorithms can predict a basic branching structure they are often used in this context [15]. Nevertheless, clustering different subsets of DNA data or adding some new samples to the data set may result in getting contradictory dendrograms. Since we are interested in getting stable dendrograms to provide a prediction of the branching structure, we tried to improve the dendrogram stability by including background knowledge.

Davidson & Ravi [9] presented an agglomerative hierarchical clustering algorithm using constraints and demonstrated the enhancement in the cluster accuracy. Their algorithm stops if no more agglomerations according to the *cannot-links* can be performed. This results in root-less dendrograms. Additionally they provided constraints from one particular partitioning and then used these constraints during the complete clustering process. In Section 2 we pointed out that this kind of constraining holds the risk of inducting wrong constraints. As a result their approach is not able to provide a rough estimation of the branching stucture on the top levels in the dendrogram and may lead to variable dendrograms.

In microarray data analysis a rough estimation of the subgrouping is best seen at the top of the dendrogram, whereas the detailed hierarchy in the lower levels is often not as important. We implemented a divisive hierarchical clustering algorithm and included background knowledge in the form of constraints. On the basis of our considerations about the effects of constraints in the hierarchical clustering process we decided to restrict the use of constraints on the first level of the dendrogram and did not to use constraints from the lower levels. By comparing the results from unconstrained and constrained divisive hierarchical clusterings we could show that this inclusion of constraints in the divisive hierarchical clustering algorithm results in more stable dendrograms. Furthermore, the positive effect of the constraints from the first level of the dendrogram even seem to improve the stability of the lower levels.

Acknowledgments

This work is supported by the Stifterverband für die Deutsche Wissenschaft (HAK) and the German Science Foundation, SFB 518, Project C5 (HAK and GP).

References

1. Demiriz, A., Bennett, K., Embrechts, M.: Semi-supervised clustering using genetic algorithms. In: Artificial Neural Networks in Engineering, New York, Troy (1999) 809–814

2. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proceedings of the 17th International Conference on Machine Learning, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2000) 1103–1110
3. Basu, S., Banerjee, A., Mooney, R.: Semi-supervised clustering by seeding. In: Proceedings of 19th International Conference on Machine Learning. (2002) 19–26
4. Klein, D., Kamvar, S., C.Manning: From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In: Proceedings of 19th International Conference on Machine Learning. (2002) 307–314
5. Bilenko, M., Mooney, R.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2003) 39–48
6. Cohn, D., Caruana, R., McCallum, A.: Semi-supervised clustering with user feedback. Technical report, Cornell University (2003)
7. Xing, E., Ng, A., Jordan, M., S.Russell: Distance metric learning, with application to clustering with side-information. In: Advances in Neural Information Processing Systems 15. (2003) 505–512
8. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning distance functions using equivalence relations. In: Proceedings of 20th International Conference on Machine Learning. (2003) 11–18
9. Davidson, I., Ravi, S.: Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In: Knowledge Discovery in Databases: 9th European Conference on Principles and Practice of Knowledge Discovery in Databases. Volume 3721 of Lecture Notes in Computer Science., Springer (2005) 59–70
10. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: Proceedings of 18th International Conference on Machine Learning. (2001) 577–584
11. Bilenko, M., S.Basu, Mooney, R.: Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the 21st International Conference on Machine Learning. (2004)
12. Buchholz, M., Kestler, H., Bauer, A., Böck, W., Rau, B., Leder, G., Kratzer, W., Bommer, M., Scarpa, A., Schilling, M., Adler, G., Hoheisel, J., Gress, T.: Specialized dna arrays for the differentiation of pancreatic tumors: A solution for a common diagnostic dilemma. Clin Cancer Res **11** (2005) 8048–8054
13. A.K.Jain, Dubes, R.: Algorithms for Clustering Data. Prentice Hall, New Jersey (1988)
14. Rand, W.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association **66** (1971) 846–850
15. Eisen, M., Spellman, P., Brown, P., Botstein, D.: Cluster analysis and display of genome-wide expression patterns genetics cluster analysis and display of genome-wide expression patterns. PNAS **95** (1998) 14863–14868