University of Bremen
Faculty of Computer Science and Mathematics

Diploma thesis

Hierarchical clustering of sensorimotor features

Konrad Gadzicki

November 10, 2008

Primary supervisor:	Prof. Dr. Kerstin Schill
Secondary supervisor:	Dr. Thomas Barkowsky

Acknowledgments

Many thanks to my family.

Abstract

This work deals with the clustering of sensorimotor features which can serve as a represention of visual scenes (images). The approach used here is to obtain a feature frequency vector by training a Self-Organizing Map (SOM) and counting the mappings of scene features to SOM output layer. Hierarchical clustering with agglomerative methods will be applied afterwards to generate a hierarchy.

Contents

CHAPTER 1

Introduction

One of the most striking aspects of human vision is the fact that the human eye has only a tiny spot, the fovea, with a high resolution which enables us to perceive our environment in detail. By moving the eyes, we are able to fixate different spots in the environment in rapid succession. This system of fixation and saccade enables to perceive and recognize the world around us[1]. It seems natural that computer science tries to mimic and reproduce this abilities in computer vision. Some research has been done on building systems which tackle the pattern recognition problem of scene by the means of limited local, foveal-like sensory input in combination with saccade-like motor actions. One such system and its sensorimotor features used for representation is used in this work as a data provider

Having a set of patterns at hand, whatever the representation may be, the question of how to structure them often arises. Are there any patterns which should be grouped together since they have a certain similarity? This last sentence already contains several concepts that have to be approached first. For the actual task of grouping, clustering methods have been developed and used in virtually every discipline in science in the course of time. A central aspect is how the output structure should be like in general, either a flat set or a hierarchical organization of nested clusters. A hierarchical structure can be desired since it delivers different levels of granularity.

With regard to this work, both representation of patterns by sensorimotor features and hierarchical clustering are relevant. Brought down to the point, the problem to be solved can be stated as

[1]Of course there is also peripheral vision which also plays a role.

Given a set of objects each represented by a set of sensorimotor features (feature - action - feature), obtain a hierarchical structure which organizes the objects.

Structure

The first two chapters cover the basics by introducing similarity measures, hierarchical clustering methods and Self-Organzing Maps. Based on these concepts and methods, in chapter four a method is derived which aims at clustering of patterns represented as sensorimotor features. Chapter five covers the testing and application of it to images. Existing approaches are introduced in chapter six. The final chapter discusses the approach.

CHAPTER 2

Clustering and similarity

This chapter covers the basics used in this work and gives a theoretical foundation.

First basic notions and concepts such as data types, patterns and similarity are covered. Next clustering in general is described followed by a more detailed look at hierarchical clustering methods. Self-Organizing Maps are presented in the next chapter.

2.1 Patterns, classes and features

2.1.1 Classes

Classes are *categories* of objects associated with some *concept* or *prototype* (de Sá, 2001). The concept of a class is a generalization of the members associated with it. It can be, for instance, described by common features shared by all class members. Prototypes are specific class members representing the class and thus all (possible) objects associated with it.

In order to describe a class by concept, one might state possible values or ranges of values for each feature. For description by prototypes, one would one or more prototypes having the possible and relevant combinations of feature values in order to represent the class.

The class space can be very often interpreted in a hierarchical way. Classes can be grouped to superclasses according to broader concepts and thus fewer common features for example. An ontology is prime example for such a hierarchical structure of classes and concepts. Figure 2.1 shows as very simple example of a hierarchy of vehicles. Here some vehicles are grouped according to the number of wheels.

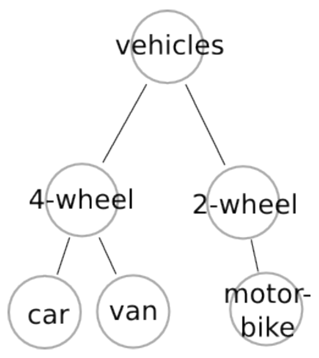


Figure 2.1: Simple example of a hierarchy of vehicles

2.1.2 Patterns

Patterns are representations of objects, also referred to as objects, cases, samples, instances. Very often, a pattern represents some real-world (physical) object, even though it is not necessarily so. The patterns used in this work are mainly images.

2.1.3 Features

Features are attributes or measurements obtained from patterns and used to describe or characterize the objects.

It should be noted that features are not necessarily raw data obtained from objects. An 512x512 pixels image is basically a set of 262.144 light intensities, but this full set in its raw form would probably never be used for clustering purpose. Instead a feature extraction step is applied beforehand in order to obtain significant features which involves transformation of the raw data with signal and image processing techniques in the case of images for instance (de Sá, 2001).

2.1.4 Types of data

The type of a feature or attribute is first and foremost depending on the type of measurement. The taxonomies in use vary to some degree. The only common point between all seems to be the distinction between numeric and nominal data. The taxonomy provided by Witten and Frank (2005) distinguishes between

- nominal
- ordinal
- interval

- ratio

data. A broad classification distinguishes between numeric data (subsumming interval and ratio) and nominal data (regarding ordinal as nominal). Numeric attributes are also referred to as continuous, nominal are also called categorical (Witten and Frank, 2005; Pedrycz, 2005).

Nominal attributes have discrete values which are taken from a fixed (and often small) set of states. Each value is rather a symbol of its own which can be compared to other ones from the same set. But there is no order (and thus no distance) and no similarity defined. Thus the only valid comparison is equality. A good example for this kind of data is the gender attribute with `male`, `female` and `hermaphrodite`[1] as values. Binary data also fall into that category being a special case with two distinct states only.

Ordinal features share the same properties of nominal attributes, but have furthermore an ordering. For instance, temperature in categorical way can be described as `hot`, `warm`, `mild` and `cold`. There is clearly an order here [2]. Still there is no notion of distance defined since it is not clear how far "apart" each pair of values is from each other. The relation between values can be evaluated with regard to equality as well as greater/smaller than (Witten and Frank, 2005). According to Pedrycz (2005), this sort of data does have a similarity measure defined which actually implies that there is a relative between individual states. I think his notion corresponds rather to the type of data described next.

Interval data are defined on a numerical scale, measured in fixed (and equal) intervals. There is an order defined on these values as well as distance. In comparison to the temperature example given above with symbolic values like `cold`, here the temperature would be measured in intervals of 10°C for instance. Particular values would then cover 0°-9°, 10°-19° and so on. Clearly one can calculate the distance between two given intervals (Witten and Frank, 2005).

Ratio data take real values and have furthermore a inherent zero point (Witten and Frank, 2005). They can be expressed by continuous or discrete values.

The taxonomy given by Gowda and Diday (1991)(cited according to Jain et al., 1999) distinguishes between qualitative and quantitative data on a broad level:

- Quantitative features
  - continuous values

[1]Just in case, we need to state a snail's gender.

[2]Whether the order is ascending or descending does not matter.

    - discrete values
    - interval values

- Qualitative features
    - nominal or unordered
    - ordinal

Quantitative features can be compared with numerical measures while the qualitative cannot. Ratio values from the classification above are split into continuous and discrete.

Impact of data type The data type has a major impact on the methods which can be applied to measure similarity between sets of patterns. Intuitively nominal data which can only be compared with regard to equality, supports only measures which compare members of distinct sets pairwise for equality. That basically means that set operations (intersection, union etc.) are the foundation.

Quantitative data on the other hand support arithmetic operations in general.

2.2 Similarity measures

The measurement of similarity plays a key role in the clustering task. All algorithms naturally require a some value for comparison of individual pattern. The question is how to obtain such a measure from the features representing the patterns.

Whether one talks about *similarity* or *dissimilarity* of objects does not really matter since both terms are two sides of the same coin. Having a formal definition of either of them, the other one can be obtained with a suitable transformation. In order to reduce the confusion of distinguishing between similar and dissimilar all the time, the term *proximity* is used to denote either of the them.

A proximity index denoted as $d(x_i, x_j)$ with x_i and x_j being patterns must satisfy the following properties namely reflexivity (2.1) and (2.2), symmetry (2.3) and non-negativity (2.4) (Jain and Dubes, 1988):

1. a) For similarity:

$$d(x_i, x_i) \geqslant \max_j d(x_i, x_j), \quad \text{for all } i \tag{2.1}$$

   b) For dissimilarity:

$$d(x_i, x_i) = 0, \quad \text{for all } i \tag{2.2}$$

2.
$$d(x_i, x_j) = d(x_j, x_i), \quad \text{for all } i, j \tag{2.3}$$

3.
$$d(x_i, x_j) \geqslant 0, \quad \text{for all } i, j \tag{2.4}$$

In order to satisfy being a metric, additional properties have to be satisfied (identity (2.5) and triangular inequality (2.6)) (Jain and Dubes, 1988):

4.
$$d(x_i, x_j) = 0, \quad \text{iff } x_i = x_j \tag{2.5}$$

5.
$$d(x_i, x_j) \leqslant d(x_i, x_k) + d(x_k, x_j), \quad \text{for all} i, j, k \tag{2.6}$$

Failing metric properties If property 3 fails (2.5), the result is a *pseudometric*. A distance of 0 does not mean necessarily that two samples are identical, but only similarity or equivalent. The cosine proximity introduced later is such a pseudometric.

If property 2 fails (2.3), the proximity measure is a *quasi-metric*. This is for instance the case for distance in a directed graph, or more practical, cities with one-way streets where obviously the distances are not necessarily equal form A to B and from B to A (Clarkson, 2006).

2.2.1 Numeric data

The numeric data type describe above (see 2.1.4, page 14) posses an inherent distance between the individual values. The similarity measures used for this kind of data are thus often making use of this fact.

2.2.1.1 Minkowski based (L_p-Norm)

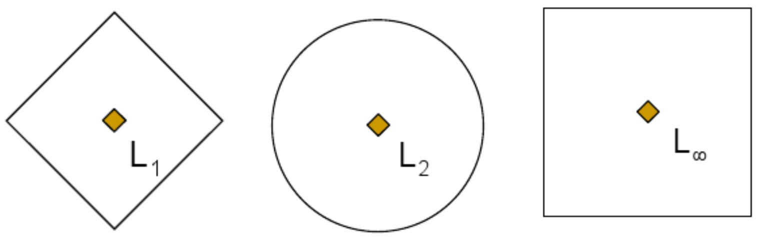
For numerical data, very often L_p metrics are used as a standard proximity function which is the Minkowski distance of order p (2.7)

$$d_p(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{2.7}$$

where x and y are patterns and n is the number of features. Deriving from the *Minkowski* norm are *Manhattan*[3] distance (L_1) ,

$$d_1(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^1 \right)^{\frac{1}{1}} = d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.8}$$

[3]also called city block distance, or taxi cab distance



(a) Manhattan metric (b) Euclidian metric (c) Maximum (infinity) metric

Figure 2.2: 2D space covered by Manhattan metric (2.2a), Euclidian metric (2.2b) and Maximum (Infinity) metric (2.2c) (`source:` Zezula et al. (2005))

Euclidian distance (L_2)

$$d_2(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^2\right)^{\frac{1}{2}} = \sqrt[2]{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.9}$$

and *maximum* or *infinity* distance (L_∞).

$$d_\infty(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^2\right)^{\frac{1}{\infty}} = \max_{1 \leq i \leq n} |x_i - y_i| \tag{2.10}$$

Figure 2.2 illustrates the room covered by the metrics above in 2-dimensional space. All L_p-based metrics cover rather compact space around vector, with the Euclidian distance spreading equally into every direction while all the other have a certain preference. They are invariant to translations in feature space but not to linear transformations in general. Euclidian metric is furthermore invariant to rotation. It should be noted that often a normalization of the input data is necessary since L_p metrics are not invariant to scaling, i.e. large values of features dominate the result (Duda et al., 2001).

Thus the Euclidian metric is probably the closest to human perception of proximity since it reflects the "natural" distance in 2D and 3D environment.

Being a measure of dissimilarity with distances ranging from 0 to ∞, a suitable transformation for similarity is $similarity = \exp(-(distance)^2)$ relating the squared error loss function to the negative log-likelihood for a Gaussion model (Ghosh and Strehl, 2006), so that a normalized Euclidian similarity is expressed by

$$s(x_i, x_j) = e^{-\|x_i - x_j\|^2} \tag{2.11}$$

2.2.1.2 Cosine

The cosine measure is especially popular in document clustering. The similarity is expressed by the cosine of the angle between two vectors and measures similarity directly. It is given by

$$s(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\|_2 \cdot \|x_j\|_2} \tag{2.12}$$

where $x_i \cdot x_j$ is the *dot*-product and $\|x\|_2$ is the Euclidian norm (p-norm with $p = 2$) of the vector.

Note that the cosine measure is not a metric since it violates the identity property (2.5). It returns a distance of 0 not only if two data points have exactly the same values, but also if the relative distribution of values is equal. It does not depend on the length of a vector but on the direction only. This leads to the high popularity of this measure for document comparison. Documents, represented by the amount of terms found in it, having the same relative distribution of terms can be treated as identical while the absolute count of terms does not matter. This property makes it possible to compare documents of different size effectively (Ghosh and Strehl, 2006).

The cosine similarity is invariant to rotation and dilation, but not to translation (Duda et al., 2001).

2.2.1.3 Extended Tanimoto or extended Jaccard

The *Tanimoto* similarity captures the degree of overlap between two sets and is computed as the number of shared attributes. One might notice that this only applies to binary data. Still it can be extended to continuous space as

$$s_{Tanimoto}(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\|^2 + \|x_j\|^2 - x_i \cdot x_j} \tag{2.13}$$

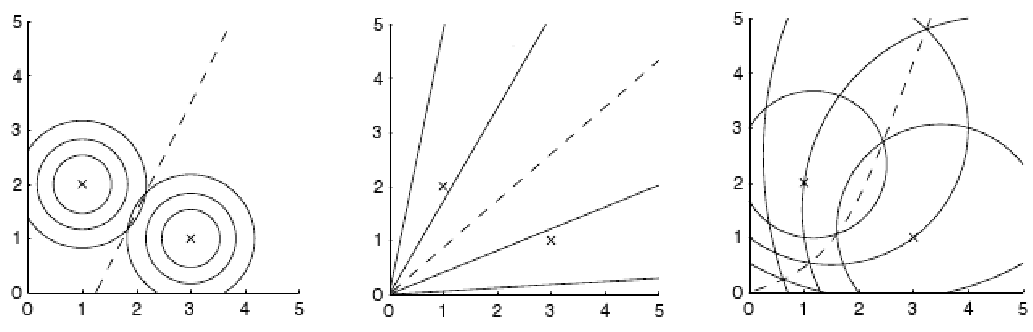
with $x_i \cdot x_j$ being the *dot*-product and $\|x\|_2$ is the Euclidian norm (p-norm with $p = 2$) of the vector. This measure is also referred to as *Tanimoto coefficient* (*TC*).

The Tanimoto distance is defined as

$$d_{tanimoto}(x_i, x_j) = 1 - s_{Tanimoto}(x_i, x_j) \tag{2.14}$$

which means nothing but subtracting the *Tanimoto coefficient* from 1. It has been shown that this measure is a metric. According to (Clarkson, 2006), it has been proofed by Deza and Laurent (1997), Spaeth (1980), Xu and Agrafiotis (2003) and Charikar (2002).

The Tanimoto coefficient shares properties from Euclidian and cosine similarity measures which will be shown next.



(a) Euclidian-based (b) cosine-based (c) Tanimoto-based

Figure 2.3: Properties of Euclidian (a), cosine (b) and extended Tanimoto (c) similarity measures. The lines plotted show the similarity for $s = 0.75$, 0.5 and 0.25. (`source:` Ghosh and Strehl (2006))

2.2.1.4 Comparison of Euclidian, cosine and Tanimoto similarity

Figure 2.3 illustrates the isosimilarity surfaces in 2D-space covered by Euclidian, cosine and Tanimoto similarity measures.

The Euclidian metric isosimilarities are concentric (hyper)spheres around a given point with similarity 1. The cosine measure produces (hyper)cones around a line through a given point and the origin. All points on this line are considered having similarity 1 with the given point. Finally Tanimoto measure isosimilarities are nonconcentric (hyper)spheres around a given point with one point having a similarity of 1. The radius of the spheres increases with the distance from the origin so that longer vectors are treated more tolerantly in comparison to Euclidian. Having a single point of similarity 1, Tanimoto behaves like Euclidian for a similarity towards 1 while for $s \to 0$, it behaves more like the cosine. Thus it shares properties of both, Euclidian and cosine, making it a rather robust, all-purpose measure (Ghosh and Strehl, 2006).

2.2.2 Binary data

Binary data consist of nominal values with only two distinct states (see 2.1.4, page 15). Similarity between binary data is computed on matching and mismatching attributes, or in other words, the equality of values. The four combinations shown in 2.1 are further referred to with the letters given in the table.

The binary measures shown in equations (2.15) to (2.19) are some of the more popular measures.

$$s_{Hamming}(x, y) = \frac{b + c}{a + b + c+} \tag{2.15}$$

		Variable 2	
		1	0
Variable 1	1	a	b
	0	c	d

Table 2.1: Index letters for binary matches

$$s_{Rand}(x,y) = \frac{a+d}{a+b+c+d} \tag{2.16}$$

$$s_{Rao}(x,y) = \frac{a}{a+b+c+d} \tag{2.17}$$

$$s_{Tanimoto}(x,y) = \frac{a}{a+b+c} \tag{2.18}$$

$$s_{Czekanowski}(x,y) = \frac{2a}{2a+b+c} \tag{2.19}$$

2.3 Clustering

Clustering is the task of discovering meaningful structures in a data set by grouping samples or observations into *clusters* of similar objects. It is data-driven in the sense that there is usually no prior knowledge about the underlying structure of the data-set and it is thus a *unsupervised* learning method.

A cluster serves as a *class* representation. Those classes are supposed to represent a “natural” structure of the data set. The clustering task is to find clusters which (hopefully) fit the natural underlying classes.

The basic idea behind probably any approach to finding these clusters is that objects belonging to the same class share similar *features* or *attributes*. The objects grouped within such a cluster are supposed be more similar to each other than to objects put into other clusters. Thus clustering aims at maximizing the intra-cluster similarity of objects while minimizing the inter-cluster similarity.

A general problem is the number of clusters or classes generated. Many algorithms require the number a priori which might cause a problem if this number is not known at all. Still depending on the application, this problem is smaller than it seems. Clustering of consumer data bases might for example only require separating the good from the bad, thus a fixed number of clusters would be fine. Other algorithms are able to approximate the number of classes. Still this is usually done by providing a sort of threshold which indicates when to generate additional clusters [4].

[4]e.g. the inter-class similarity exceeding a certain value

2.3.1 Clustering vs classification

Even though both task basically do the same (map a set of pattern onto a set of classes), clustering and *classification* have a significantly different purpose.

Classification works with *labeled* data, usually in a *supervised* way. That means, for a given set of samples, the underlying class of each sample is known from the start. The learning step during classification aims at obtaining a classifier which is able to map an object to its corresponding class correctly.

Clustering on the other hand works with unlabeled data from unknown classes. Often even the number of classes is not known. Just as stated above, it aims at separating the data set into a (given) number of clusters according to similarity of the objects.

2.3.2 Steps in the clustering process

According to Jain and Dubes (1988), the typical steps in the clustering process are

1. pattern representation (optionally including feature extraction and/or selection)
2. definition of a pattern proximity measure appropriate to the data domain
3. clustering or grouping
4. data abstraction (if needed)
5. assessment of output (if needed)

Figure 2.4 shows the first three of these steps. *Feature extraction* refers to the transformation of input features into new meaningful features [5] while *feature selection* means the selection of a relevant subset of features.

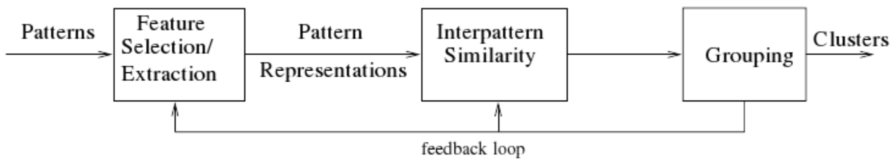


Figure 2.4: Steps in the clustering process (`source:` Jain et al. (1999))

Pattern similarity is usually measured by some distance function. Section 2.2 on page 16 deals further with this topic.

The grouping step finally performs the generation of clusters by dividing or merging (sub)sets of patterns according to the previously defined proximity

[5]e.g. the transformation of raw image data with signal processing techniques

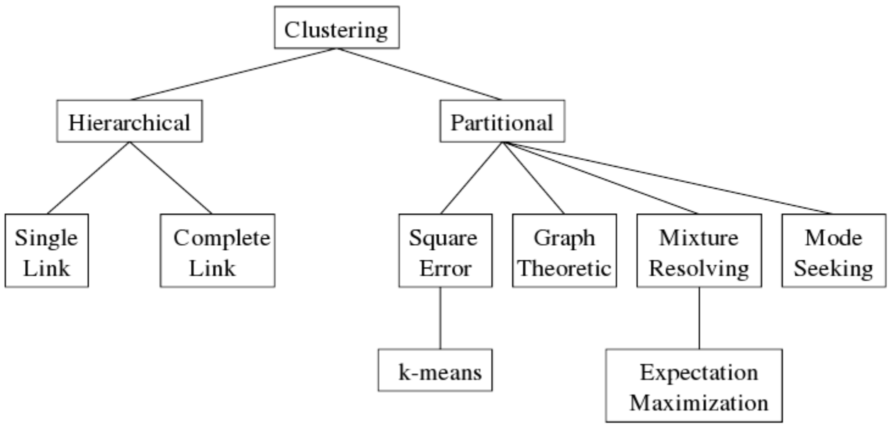


Figure 2.5: Taxonomy of clustering approaches (`source:` Jain et al. (1999))

measure. The membership of individual patterns can be either crisp (each pattern belongs to a particular cluster) with the effect of a hard partitions or it can be fuzzy with each pattern having a certain degree of membership to each cluster. Furthermore partitions can be structured hierarchically with nested clusters or flat.

2.3.3 Taxonomy of clustering approaches

Since clustering, being a powerful tool for data exploration, is utilized by various disciplines for various applications, the number of clustering algorithms is legion. A crisp categorization of all existing approaches is thus difficult, if not impossible to obtain, since there is often an overlap between them.

Figure 2.5 shows a classification based on the discussion in Jain and Dubes (1988).

Berkhin (2002) provides another taxonomy.

- Hierarchical Methods
  - Agglomerative
  - Divisive
- Partitioning Methods
  - Relocation Algorithms
  - Probabilistic Clustering
  - K-medoids Methods
  - K-means Methods
  - Density-Based Algorithms
    - Density-Based Connectivity Clustering

        * Density Functions Clustering

- Grid-Based Methods
- Methods Based on Co-Occurrence of Categorical Data
- Constraint-Based Clustering
- Clustering Algorithms Used in Machine Learning
    - Gradient Descent and Artificial Neural Networks
    - Evolutionary Methods
- Scalable Clustering Algorithms
- Algorithms for High Dimensional Data
    - Subspace Clustering
    - Projection Techniques
    - Co-Clustering Techniques

The classical discrimination is between hierarchical and partitioning algorithms. Hierarchical construct nested partitions while partitioning algorithms produce only one, flat set of partitions.
Jain et al. (1999) state furthermore some properties which generally speaking may apply to all approaches.

Agglomerative vs. divisive Agglomerative approaches start with singleton clusters and merge those successively into larger clusters until a stopping criterion is reaches. Divisive methods on the other hand start with one cluster for all patterns and perform splitting until a stopping criterion is satisfied.

Monothetic vs. polythetic This refers to the sequential or simultaneous use of features, i.e. whether all features of a pattern are used simultaneously for comparison (polythetic) or in one by one.
The usefullness of monothetic partitioning is somewhat questionable. A first, broad division of the data set by single features like, for instance, gender might be fine depending on the purpose, but performing the entire clustering based on that principle would probably result in a highly fragmented set of clusters with limited information value.
Most algorithms are polythetic.

Hard vs. fuzzy Hard algorithms assign patterns to crisp clusters during operation and in its output. Fuzzy methods assign for every pattern a degree of membership to every cluster.

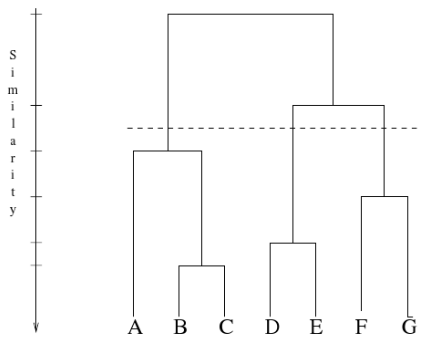


Figure 2.6: Example of a dendrogram. (`source:` Jain et al. (1999))

Deterministic vs. stochastic Basically means whether the assignment of patterns to clusters follows a specific, deterministic way or is based on random drawing of assignments from possible sets of partitions. Mostly relevant for methods aiming at minimizing the square error of partitions like *k-means*.

Incremental vs. non-incremental This property is relevant if the pattern set is extremely large, and constraints on execution time or memory limitations have to be considered. This issue has more or less risen together with data mining techniques which pushed the development of incremental methods which aim at minimizing the number of scans of the pattern set, reduce the number of pattern examined or the size of data structures used.

Out of these vast approaches, only hierarchical clustering will be covered next. The partitioning methods have relatively low relevance for this work.

2.4 Hierarchical clustering

Hierarchical clustering methods generate a hierarchy of clusters also referred to as a *dendrogram*. The result is thus a tree of nested clusters. Each tree node represents a cluster and contains children which are clusters as well. The original patterns are leafs of the tree. Figure 2.6 illustrates a dendrogram.

Such a hierarchy can be evaluated at different levels of granularity. In order to obtain a flat set of clusters, the tree can be cut at a specific threshold, followed by merging the siblings into the top-level clusters if appropriate.

Agglomerative variants start with singleton clusters containing a single pattern only. In subsequent steps two or more clusters are merged into larger clusters at each time step. The algorithms stops if a certain criterion like a given number of clusters is reached or all clusters are merged into one. Divisive variants start with one single cluster and partition the data set until a stopping criterion is met or only singleton clusters remain.

Divisive approaches will not be covered further. Basically one can take any partitioning algorithm and apply it recursively on the clusters obtained in every step. The agglomerative approaches will be dealt with next.

2.4.1 Agglomerative hierarchical algorithm

The merging algorithm consists of the following steps (de Sá, 2001):

1. Given n patterns, place the pattern in singleton clusters such that the number of clusters $c = n$
2. While $c \geqslant 1$[6]
    a) Determine the two nearest clusters using an appropriate similarity measure and linkage rule
    b) Merge the two nearest clusters, thus reducing the number of clusters
    c) Decrease c

Obviously the results depend both on the similarity measure and the linkage rule used. Similarity measures will be covered later in section 2.2, page 16. Here it refers to the similarity between patterns. Regarding linkage rules the most popular are *single linkage*, *complete linkage*, *average linkage* rules and *squared-distance linkage* (Ward's method). Linkage rules define the similarity or distance between clusters.

2.4.1.1 Single linkage

Single linkage defines the distance between two clusters as the minimum distance all pairs of patterns, one from each cluster. It is therefore a nearest neighbor based method. This method was introduced by Sibson (1973), and is considered a classical approach found in any textbook.

$$d(w_i, w_j) = \min_{\mathrm{x} \in w_i, \mathrm{y} \in w_j} \|\mathrm{x} - \mathrm{y}\| \tag{2.20}$$

$\|\mathrm{x} - \mathrm{y}\|$ is some distance norm; very often Euclidian norm is used as sort of standard distance measure.

This method suffers from bridge building. i.e. it connects structures which are linked by a chain of pattern as depicted in figure 2.7a. Still it is suitable if the data set has a filamentary shape as seen in figure 2.7b.

[6]This is the most simple stopping criterion making the algorithm run until only one cluster is left. Other criteria are possible of course

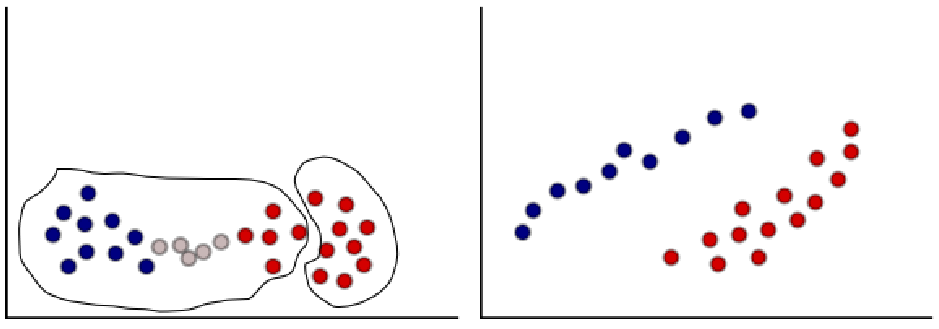


Figure 2.7: (a) Bridge effect: Parts of the red patterns are grouped with blue ones due to some noisy patterns (gray) in between. (b) Filamentary data sets which can be separated well with single linkage.

2.4.1.2 Complete linkage

Complete linkage uses the maximum distance between all pairs of pattern, one from each cluster, and can be characterized as a *furthest neighbor* method (Defays, 1977).

$$d(w_i, w_j) = \max_{\mathrm{x} \in w_i, \mathrm{y} \in w_j} \|\mathrm{x} - \mathrm{y}\| \tag{2.21}$$

This works well on compact, globular data and tends to form sphere-like clusters, but fails to cluster filamentary, chain-like sets. Figure 2.8 shows the application to compact data with a bridge connecting the sets. Still it should not be used if one expects lots of noisy patterns in the data set since outliers tend to be favored. For good results, it requires the underlying clusters to be well-separated in space.

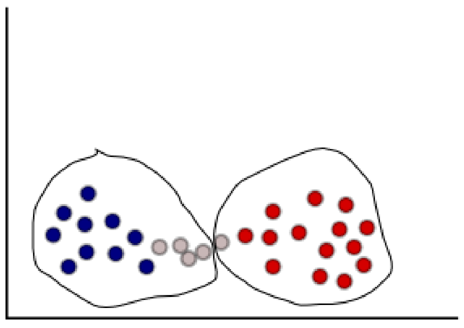


Figure 2.8: Compact data clustered by complete linkage

Single and complete linkage are the two extremes of how to define the similarity of clusters in terms of hierarchical clustering. They disregard the actual set of patterns already present in a cluster and evaluate maximum or minimum values only. Nevertheless they work well if the data set meets the requirements.

Average methods presented next evaluate all patterns present within a cluster. Therefore they also work on atypical data sets.

2.4.1.3 Average linkage between groups

This linkage rule assess the average distance between all pairs of pattern with each pattern being in distinc clusters (Voorhees, 1986). This method is really in between single and complete linkage. It does not suffer neither from the chaining effect nor from preferring outliers which makes it popular as a general linkage rule.

$$d(w_i, w_j) = \frac{1}{n_i n_j} \sum_{\mathrm{x} \in w_i} \sum_{\mathrm{y} \in w_j} \|\mathrm{x} - \mathrm{y}\| \tag{2.22}$$

2.4.1.4 Average linkage within groups

Being a variant of the previous one, the distance between clusters is the average distance of all possible pairs of pattern if they formed a single cluster. This method is aiming at minimizing the within-cluster average error.

$$d(w_i, w_j) = \frac{1}{n_i + n_j} \sum_{\mathrm{x},\mathrm{y} \in \{w_i, w_j\}} \|\mathrm{x} - \mathrm{y}\| \tag{2.23}$$

2.4.1.5 Ward's method

Proposed in Ward (1963), this method aims at minimizing the intra-cluster variance. It thus computes the inter-cluster distance by summing the squared within-cluster distances of all patterns to a cluster centroid for hypothetical clusters.

$$d(w_i, w_j) = \frac{1}{n_i + n_j} \sum_{\mathrm{x} \in \{w_i, w_j\}} \|\mathrm{x} - \mathrm{m}\| \tag{2.24}$$

$\mathbf{m}$ is the centroid of the resulting cluster.

2.4.1.6 Lance-Williams

The *Lance-Williams* formula is a general expression describing various agglomerative clustering approaches. Equation (2.25) shows the formula which express the distance between cluster C and another cluster formed by A and B. d is a distance function

$$d_{A \cup B, C} = \alpha_A d_{A,C} + \alpha_B d_{B,C} + \beta d_{A,B} + \gamma |d_{A,C} - d_{B,C}| \tag{2.25}$$

Depending on the parameters α_A, α_B, β and γ, several agglomerative clustering methods can be expressed. Table 2.2 shows the parameters for some clustering methods. m_A, m_B and m_C are the number of points in the cluster.

Clustering Method	α_A	β	γ
Single linkage	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete linkage	$\frac{1}{2}$	0	$\frac{1}{2}$
Group average	$\frac{m_A}{m_A+m_B}$	0	0
Ward's	$\frac{m_A+m_C}{m_A+m_B+m_C}$	$\frac{-m_C}{m_A+m_B+m_C}$	0

Table 2.2: Lance-Williams parameters

2.4.1.7 Transitive Closure method

Ohsumi and Moroi (1975) investigated how a dendrogram as a solution of a hierarchical clustering process and fuzzy relations are connected. Their findings showed that computing the transitive closure of a similarity matrix generates a hierarchical structure and satisfies a fuzzy relation.

Without diving into the depths of fuzzy set theory (Zadeh, 1965), a fuzzy relation has to satisfy reflexivity, symmetry, max-min transitivity and min-max transitivity. A similarity matrix containing all pairwise similarities of patterns satisfies symmetry and reflexivity, if the similarity measure was obtained with an appropriate measure like a metric. So the open question is transitivity which cannot be assured. But it is possible to compute the transitive closure of a matrix and thus satisfy this condition as well (Dubois and Prade, 1980).

The transitive closure matrix (TC matrix) obtained can be evaluated by going through the thresholds in descending (ascending) way and merging (splitting) the clusters at a given threshold. The result is a dendrogram.

2.4.1.8 Divisive hierarchical clustering

The divisive method is more or less an inverted agglomerative approach. Instead of starting with singleton clusters and merging them until all only one cluster is left, this algorithm starts with a single cluster and splits it into two. This is done iteratively for all clusters until only singleton clusters remain.

This method is seems to be hardly used at all. It produces the very same results as the merging approach but is computationally heavy. In order to divide a cluster, basically all possible subsets have to be analyzed.

2.4.2 Noteable hierarchical clustering developments

Much research has been conducted to improve the performance (quality and/or time and space requirements) of clustering methods compared to classical approaches. Some noteable methods with regard to hierarchical agglomerative clustering are presented next.

COBWEB (Fisher, 1987) is a method for conceptual or model-based clustering, producing a dendrogram which can be seen as a classification tree. Each cluster in the tree stores the conditional probabilities for attribute-concept pairs which are basically concept-specific Naïve Bayes classifiers. New patterns can be inserted iteratively according to a heuristic called *category utility*.

CLASSIT (Gennari et al., 1989) is similar to COBWEB, though it works with numerical values, and associates normal distributions to nodes.

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) (Zhang et al., 1996) is a clustering method which is suitable for very large databases by being able to operate with limited resources (memory, time constraints). It utilizes a height-balances tree which stores nonleaf nodes containing data summaries, called Clustering Features (CF), instead of the original data. Leaf nodes also store a CF but have a threshold requirement (a radius or diameter) in addition which governs whether new entries can be summed up by a particular leaf or have to constitute a new leaf.

CURE (Cluster Using REpresentation) (Guha et al., 1998) is a hierarchical agglomerative clustering method. It represents a cluster by a fixed number of points which are supposed to to capture the shape of a cluster. This is a middle-ground between the representation by a centroid and all points. Furthermore it employs sampling from the input data instead of always using the entire set in order to be applicable to very large databases. The number of clusters has to be given beforehand.

ROCK (Robust Clustering algorithm for Categorical Data) (Guha et al., 1999) by the same developers as CURE has a lot in common with the before-mentioned method, but was developed for categorical data. It defines a link between two data points as the number of common neighbors, given a certain threshold as a minimum.

CHAMELEON (Karypis et al., 1999) is a two phase hierarchical clustering algorithm. First it uses graph partitioning algorithm to cluster the input data into rather small clusters and obtains a sparse cluster representation using a k-nearest neighbor graph. In a second step, it builds a hierarchy with agglomerative methods by the terms of relative inter-connectivity and relative closeness.

Hierarchical density-based clustering (Ankerst et al., 1999)

Hierarchical subspace clustering (Achert, 2007)

2.5 Evaluation of clustering results

When it comes to the evaluation of results, it is rather hard to assess the quality of a hierarchy generated. A lot of the trouble comes from the fact that any evaluation has to be based on subjective measures.

2.5.1 Ugly duckling theorem

The ugly duckling theorem (Watanabe, 1969) states that there is not objective measure of similarity. It cannot be distinguished between two classes when they are compared over all features. So any arbitrary pair of objects will be equally similar. What is necessary here is the application of domain knowledge in order to in a useful way.

This does not only apply to the similarity measures introduced before but also to the clustering methods. With regard to hierarchical agglomerative clustering it the question which clusters should be merged at a specific point.

The ugly duckling theorem is somewhat unintuitive since it takes a look from a pure mathematical point of view. It forces one to acknowledge that any apparently simple similarity definitions are based on some assumptions and only hold true under these assumptions. With regard to the evaluation of a hierarchy, the quality assessment also depends on the assumptions.

Based on assumptions about what the outcome should be, some quantitative measures have been developed, for instance the cophenetic correlation.

2.5.2 Cophenetic distance and correlation

The cophenetic correlation is a evaluation measure for hierarchical clustering. It makes use of the *cophenetic distance* which is the distance between a pair of objects given by the proximity at which a pair of objects was grouped into one cluster for the first time in the process of agglomerative clustering. For instance, when cluster A is merged with cluster B at the distance of 0.2, then the cophenetic distance of all patterns from A to all from B is 0.2. Based on these values, a cophenetic distance matrix can be drawn.

The *cophenetic correlation coefficient* (CPCC) (Sokal and Rohlf, 1962; Farris, 1969) is the correlation between the cophenetic distance matrix and the original dissimilarity matrix and is a standard measure for expressing how well a hierarchical clustering method fits the input data (Tan et al., 2005). For instance, comparing different linkage rules might be done with this coefficient in order to see which fits the input data best.

2.5.3 Precision, recall and F-measure

The three measures are classification oriented ones. Their basis is the fraction of objects from a particular class associated with a particular cluster.

Precision *Precision* states the fraction of a cluster x that belongs to a particular class c.

$$precision(x, c) = \frac{n_{x_c}}{n_x} \tag{2.26}$$

with n_{x_c} is the number of objects belonging to class c in cluster x, and n_x is the total number of objects in cluster x (Larsen and Aone, 1999).

Recall *Recall* expresses the fraction of objects from a particular class c found in a cluster x out of all objects of class c.

$$recall(x, c) = \frac{n_{x_c}}{n_c} \tag{2.27}$$

with n_{x_c} being the number of objects belonging to class c in cluster x, and n_c being the number of all objects of class c in all clusters (Larsen and Aone, 1999).

F-measure The *F-measure* is a combination of the two before-mentioned precision and recall. It states to which extend a cluster x contains only and all objects of class c (Larsen and Aone, 1999).

$$F(x, c) = \frac{2 \cdot precision(x, c) \cdot recall(x, c)}{precision(x, c) + recall(x, c)} \tag{2.28}$$

The values produced by the measure are in the range of $[0 \ldots 1]$ with 1 being the best.

The f-measure for a particular class c is given by equation ¸(2.29).

$$F(c) = \max_x (F(x, c)) \tag{2.29}$$

Basically one computes the f-measure for all clusters and picks the best value. The idea is pretty simple: if there is a cluster with a high f-measure, this particular cluster represents this class very well by consisting of these class' objects only (or nearly only in practice) as well as containing (nearly) all objects of this class. For a particular class, it thus states how well this class is represented *somewhere* among all clusters.

The overall *F-measure*, expressing the quality for the entire set of clusters, is given by equation (2.30). For each class, the f-measure is computed and weighted over the number of objects in relation to the total number and summed up.

$$F = \sum_c \frac{n_c}{n} F(c) = \sum_c \frac{n_c}{n} \max_x (F(x, c)) \tag{2.30}$$

with n_c being the number of objects in cluster c and n is total number objects.

The f-measure can be applied to flat cluster sets as well as hierarchical ones. In the case of a hierarchy, every node is a cluster, and for the computation each node has to flattened. Thus all objects from the subtree have to be considered.

CHAPTER 3

Self-Organizing Maps

Self-Organizing Maps (SOM) are based on the work of Teuvo Kohonen from the 1980s (Kohonen, 1982a), which investigate the self-organization process on a theoretical basis. The actual SOM algorithm was introduced in Kohonen (1982b), reviewed in Kohonen (1990) as well as Kohonen (1998) and described in detail in Kohonen (2001) [1].

SOMs are a kind of Artificial Neural Networks (ANN). A categorization given in Kohonen (1990) distinguishes between three kind of ANN:

Feedforward networks These transform a set of input signals into output signals. The transformation is learned in a supervised way.

Feedback networks Here the input signal defines the initial state of the system. After a series of state transitions. The final state reached after the computation is then the output signal.

Competitive or self-organizing networks The computing elements compete against each other in their activations and develop into "specialists" for specific input signals. This is done is unsupervised way; the connection strengths are adopted based on the properties of input data.

Self-Organizing Maps fall into the last category.

A SOM is able to realize a mapping of nonlinear statistical relationships between high-dimensional input vectors to a two-dimensional grid-structure. The mapping preserves most of the topological information of the input data set which is an important property for finding cluster structures in the pattern set. A node or local group of nodes with a specific location in the map corresponds

[1]Note that this is the 3rd edition of his book; first edition is from 1998

to a specific domain of input patterns (Kohonen, 1990). Similar patterns in input space will be mapped to nodes located near each other. Furthermore the map structure is able to generalize by interpolating between samples.

Basic working principles of a SOM are therefore:

- Vector quantification: a set of vectors in the output layer is quantified in order to fit the input vectors.
- Projection from input to output space: the input vectors are projected to the previously quantified vectors in output space.

Topological maps in brains The basic idea of SOMs is grounded in topologically organized structures in the brain which are spatially organized according to the sensory input (Kangas, 1994). Several of these computational maps have been discovered, "including visual maps of line orientation and direction of motion, auditory maps of amplitude spectrum and time interval, and motor maps of orienting movements."(Knudsen et al., 1987, p. 61). An overview about research results is given in Knudsen et al. (1987) and Anderson et al. (1990).

The idea of self-organizing maps and their resemblance to the self-organization process found in brains has been widely accepted. Rojas (1996) states with regard to the visual cortex which is the area responsible for the processing of inputs from the visual field that the regions of the cortex are topologically related to the inputs, e.g. neighboring regions of the cortex process neighboring regions of the visual field, making the cortex some sort of map of the visual field.

3.1 Structure of network

As mentioned before, a Self-Organizing Map is a sort of Neural Network. It consists of an input layer and a neuron layer which is usually referred to as the map. The map nodes are arranged in a grid and have a neighborhood relation to each other.

The input layer is fully connected with the map layer and each input node corresponds to a feature of a pattern. Basically the input layer corresponds to the feature vector of a pattern. Each map neuron has a weight vector of the size of the input vector; it serves as a prototype (or model or codebook). Figure 3.1 shows an example of a SOM. The map is actually the output layer of the network, even though one is actually free to introduce a third output layer which consists of desired classes only and link the map nodes to these nodes.

The activation of a map node given a pattern is defined as the distance between the input vector of the pattern and the weight vector of the node. Again, a large number of distance measures are possible here, ranging from Euclidian distance to more exotic ones.

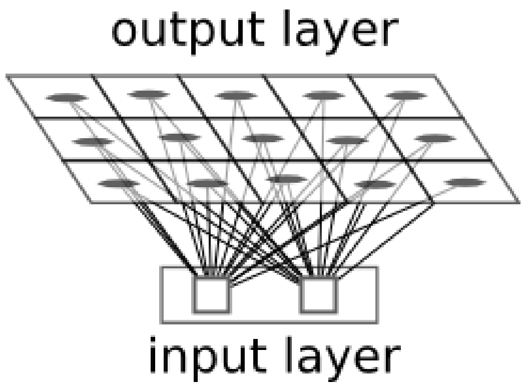
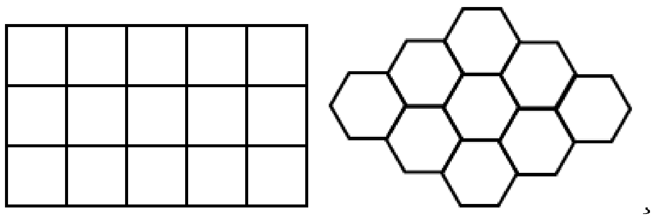


Figure 3.1: SOM illustration

3.1.1 Map grid

The map neurons are usually arranged in a 1 or 2-dimensional grid which is usually rectangular or hexagonal but extensions to higher spaces like hyperbolic space exist as well (Ontrup and Ritter, 2001). Figure 3.2 shows examples for the first two grid types.



(a) Rectanglur grid (b) Hexagonal grid

Figure 3.2: Basic grid types: (a) rectangular and (b) hexagonal.

The grid basically defines the neighborhood relation between nodes and thus also the position of each neuron in output space. In other words, it defines the topology of the output data. The distance in output space is then defined as the distance between neurons in the grid. Usually Euclidian distance is used for the 2-dimensional type.

It is also possible to wrap around the map on one or two axis, thus producing a cylinder or toroid structure as seen in figure 3.3. Wrapping is only useful if the data set supports it (circular data).

3.1.1.1 Weight vectors

Each neuron n has a associated weight vector $\mathbf{m_n} = [m_{n1}, m_{n2} \ldots m_{nd}]$ with d being the dimension of the input vector. Thus, corresponding to the feature

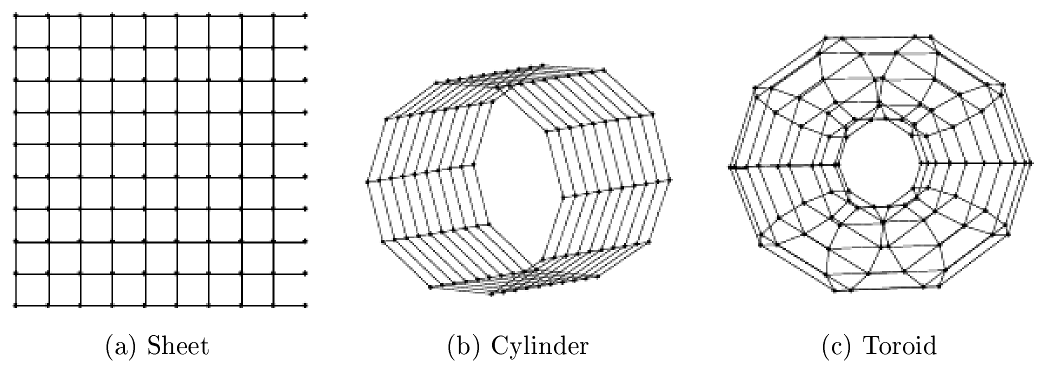


Figure 3.3: Different map shapes. (a) shows the classical map shape without wrapping, (b) and (c) depict the map topology with wrapping along one and two axis respectively (`source:` Vesanto et al. (2000))

vector of a pattern, it defines a neuron's position in input space.

These distance between a neuron and a given pattern is defined as the distance between the prototype of the former and the feature vector of the latter. Which distance measure should be used, is rather up to the properties of the representing vectors.

3.1.2 Initialization

The initialization requires the following parameters:

Map dimensions and shape The shape can be chosen from the above (see section 3.1.1, page 37) mentioned grid layouts. As a rule of thumb: the more direct neighbors a neuron has, the more possibilities there are to arrange a group of neurons while preserving the distances between them. Thus usually a hexagonal shape it recommended.

The number of neurons, determined by the map dimensions, has a high impact on the results. It should be selected as high as possible in order to allow the map to converge smoothly while maintaining its generalization capabilities. However with tens of thousands of neurons, the application computationally heavy.

Weights SOMs are very robust with respect to initialization of prototypes. One can choose among

- *random initialization:* Vector components are initialized with random values. It is not even necessary that the random values for each component are from the possible range of values from the input data, but it would speed up the convergence process and is therefore recommended.

- *sample initialization:* Prototypes are drawn randomly from the input data.
- *linear initialization:* Weights are initialized in ordered way along the principle eigenvectors of the input data. The eigenvalues can be computed by Gram-Schmidt ortogonalization, for instance.

3.1.3 Learning algorithm

The SOM is trained by patterns iteratively for a given number of times. It is suggested that the number of iterations is a multiple of the size of the data set in order to present the input patterns equally often. At least, each sample should be presented once as a minimum.

The actual learning consists of the following steps which are performed during every iteration:

1. **Finding the best matching node**
   For each map node, the distance between input vector and weight vector is computed. The node with the minimum distance is the winning unit, also referred to as Best-Matching Unit (BMU). More formally, it is defined as

   $$\|\mathbf{x} - \mathbf{m}_c\| = \underset{n}{min}\{\|\mathbf{x} - \mathbf{m}_n\|\} \tag{3.1}$$

   where $\|\cdot\|$ is the distance measure, $\mathbf{x}$ is the input vector and $\mathbf{m}_c$ is a neuron.

2. **Adaption of weights**
   The weights of (all) nodes are tuned according to the distance between weight vector and input vector, the relative distance to the winning node (neighborhood distance) and a learning rate.

   The update rule for the weight vector of neuron n is:

   $$\mathbf{m}_n(t+1) = \mathbf{m}_n(t) + \alpha(t)\ h_{c_n}[\mathbf{x}(t) - \mathbf{m}_n(t)] \tag{3.2}$$

   where t denotes time. $\mathbf{x}(t)$ is an input vector at time t, the neighborhood kernel around the Best-Matching Unit c is given as h_{c_n} and finally, the learning rate $\alpha(t)$ at a specific time.

The adjustment of weight vectors performs the actual learning and adjusts the nodes toward the input space. The individual nodes are thus shifted step by step to match patterns from the data set. This principle has been applied before in other algorithms (namely Learning Vector Quantification (LVQ) Kohonen (2001)). The trick here is that not only one matching node gets adjusted but the local neighborhood of the node as well, so that not only one node shifts toward the input signal but a local group of nodes. Figure 3.4 illustrates this.

Learning rate and neighborhood change over time. Initially the learning rate is rather high and the number of effected neurons large, too, which produces a rough global ordering. Towards the end of learning, the learning rate is small and only few neurons surrounding the winner node are effected at all, thus allowing for a fine tuning.

This produces on the long run a map which corresponds to the topology of the input data.

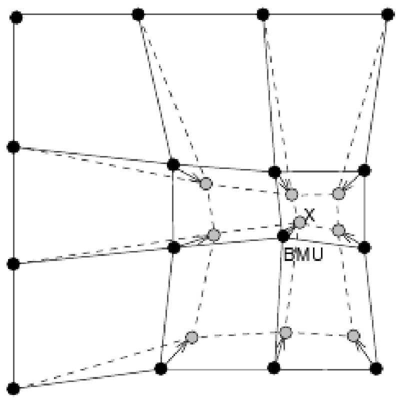


Figure 3.4: Updating the Best-Matching Unit and the local neighborhood of it. The solid lines depict the situation before updating, the dashed after. (`source:` Vesanto et al. (2000))

3.1.3.1 Learning rate

The learning rate influences how strong a node gets shifted toward a pattern. It is usually a decaying function, so that starting from given initial learning rate, the learning rate drops in the course of time. It usually depends on the current time-step out of all time-steps.

The learning rate gets applied as a factor to the distance between input vector and weight vector and therefore has to be in the range of $[0 \ldots 1]$.

The type of function is usually free to choose. It can be a linear function, piecewise defined or logarithmic function. Towards the end of the convergence process, the learning rate is constant in some applications.

3.1.3.2 Neighborhood function

The neighborhood function defines, how strong a group of neurons surrounding the Best-Matching Unit is effected during updating their vectors. It can be interpreted as the connection strengths between neurons from the map layer.

Just like the learning rate, the neighborhood changes over time and shrinks in the course of time. So if initially more or less the entire map is covered, towards the end only few neurons around the winning node are effected.

The size of the neighborhood is usually computed by a time dependent, decaying function. It defines the maximum distance in terms of *map distance* between neurons and the BMU at a given time step. Map nodes with a distance higher than the radius are not adjusted. The size is denoted as $\sigma(t)$ in the SOM literature. An example for an exponential decay function might be

$$\sigma(t) = \sigma_0 \ \exp\left(-\frac{t}{t_{total}} \ \log\sigma_0\right) \tag{3.3}$$

where σ_0 is the initial neighborhood radius (usually half of the map width or height), and t_{total} is the total number of iterations. But again, the function computing the size might be any as long as it is decaying. In some works, the neigh

The shape of the neighborhood defines the strength of adjustment within the radius around the BMU. There is a lot of possible shapes. Popular ones are *Gaussian* (3.4), *cylinder* (3.5), *cone* (3.6), *dome* (3.7) or *mexican hat* (3.8). Figure 3.5 illustrates the various shapes.

$$h_{cn}(t) = \exp\left(-\frac{\|r_c - r_n\|^2}{2\sigma^2(t)}\right) \tag{3.4}$$

$$h_{cn}(t) = \begin{cases} 1 & \text{if} \|r_c - r_n\| \leq \sigma(t) \\ 0 & \text{else} \end{cases} \tag{3.5}$$

$$h_{cn}(t) = \begin{cases} 1 - \frac{\|r_c - r_n\|}{\sigma(t)} & \text{if} \|r_c - r_n\| \leq \sigma(t) \\ 0 & \text{else} \end{cases} \tag{3.6}$$

$$h_{cn}(t) = \begin{cases} \cos\left(\frac{\pi\|r_c - r_n\|}{2\sigma(t)}\right) & \text{if} \|r_c - r_n\| \leq \sigma(t) \\ 0 & \text{else} \end{cases} \tag{3.7}$$

$$h_{cn}(t) = \frac{1}{\sqrt{2\pi}\sigma^3}\left(1 - \frac{\|r_c - r_n\|^2}{\sigma^2}\right) \exp\left(-\frac{\|r_c - r_n\|^2}{2\sigma^2(t)}\right) \tag{3.8}$$

3.2 Properties

It has been mentioned that the SOM output layer is supposed to preserve the topology of the input data. Even though one can intuitively agree with this statement for a local group of output nodes, one can doubt whether it is true for the global topology.

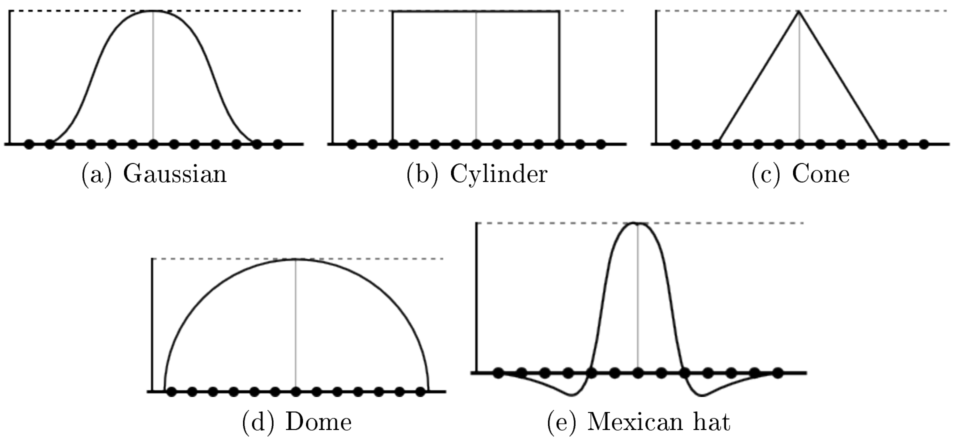


Figure 3.5: Neighborhood functions illustrated.

Ordering property The ordering property, or rather the ordering of output nodes on a global scale, has only been proofed for the 1-dimensional case (for instance Erwin et al., 1992). For the multi-dimensional case, it could only be shown for some particular cases (Flanagan, 1996) but not for the general one. This problem arises to large extend from the lack of agreement on how to order multi-dimensional data.

CHAPTER 4

Clustering sensorimotor features

This chapter covers the conceptual aspects of the clustering algorithms used which ground on the concepts and algorithms described in the previous chapters. The application to test data is described in the next chapter (chapter 5, page 61).

The clustering algorithms presented here generate a strict hierarchy where nodes located higher in the hierarchy contain sub-sets of features common to all children nodes. This work is heavily oriented towards the application to scenes (e.g. images of scenes) which shall be grouped according to common features.

The first section deals with related applications which serve as data providers and furthermore define the context of clustering by relying on the hierarchy generated. Next a specification of the input data is given. This covers the sensorimotor features as well as scene representation . This is followed by a description on how to define a similarity on these data which basically leaves the options of defining a similarity measure directly on the input data or transforming those to a more suitable format. Finally the last section deals with the hierarchy generation.

4.1 Related applications

This section introduces applications / algorithms which are related to this work. This is mainly the Okusys system for visual scene analysis which relies on a hierarchy of classes for classification of images. So far this hierarchy has been generated manually. This work proposes an unsupervised way to obtain such a hierarchy. Furthermore Okusys provides the input data for the clustering algorithm by extracting sensorimotor features from images.

4.1.1 Okusys

Okusys (Schill et al., 2001) is a biologically inspired system for recognition of images with saccadic eye movements. The insight behind Okusys is that humans do not recognize an image as a whole but by rapidly moving along small regions of interest. Each fixation at a specific region only covers a small area of a scene but by performing rapid saccadic eye-movements, all relevant regions can be covered.

Okusys mimics this behavior. First, in a preprocessing step, areas of interest are identified by linear and nonlinear filtering. Those spots or rather the foveal features located there serve as possible fixation targets. Each pair of foveal features can be connected by a saccadic eye-movement to obtain sensorimotor features which represent the scene or image. This bottom-up process is accompanied by a top-down process of adaptive reasoning about which saccade to perform in order to recognize an image. The selection strategy is based on the principle of information gain and tries to predict saccades in order to gather evidence for the current scene at hand. It utilizes a knowledge base of sensorimotor representations of scenes which has to be learned beforehand. From a cognitive point of view, the knowledge base serves as a long-term memory while the low-level processing of images (extracted features and performed saccades) correspond to the short-term memory.

The system is trained in a supervised way. All scenes are presented together with feedback.

Okusys has been already used as a visual scene analysis module in other works (Wolter, 2006; Reineking, 2008).

4.1.1.1 Feature extraction

The question which locations of a image are "interesting" from a human point of view has been addressed in Zetzsche et al. (1998). The analysis of recorded human eye-movements viewing natural scenes with higher-order statistics revealed the bias to fixate locations with frequency components of multiple orientations (e.g. curved edges). This agrees with the concept of intrinsic dimensionality. This concept relates the degrees of freedom provided by a domain to the degrees of freedom actually used by a signal and states that the least redundant (the most informative) features in images are intrinsically two-dimensional signals (i2D-signals) (Zetzsche and Krieger, 2001; Zetzsche and Nuding, 2005). Figure 4.1 illustrates the concept.

On the low-level stage, firstly nonlinear i2D-selective operators are applied to the image in order to obtain a list of fixation candidates. Secondly the feature vectors describing the characteristics of fixation locations are generated by combining the outputs of linear orientation selective filters (Schill et al., 2001). Figure 4.2 shows a unfiltered and filtered picture.

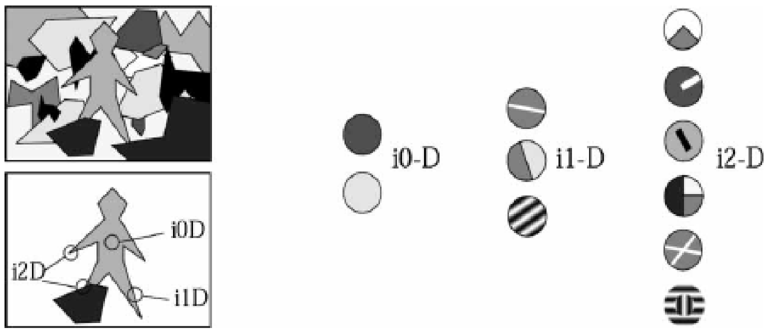


Figure 4.1: Local signals with different intrinsic dimensionality. (`source:` Zetzsche and Nuding (2005))

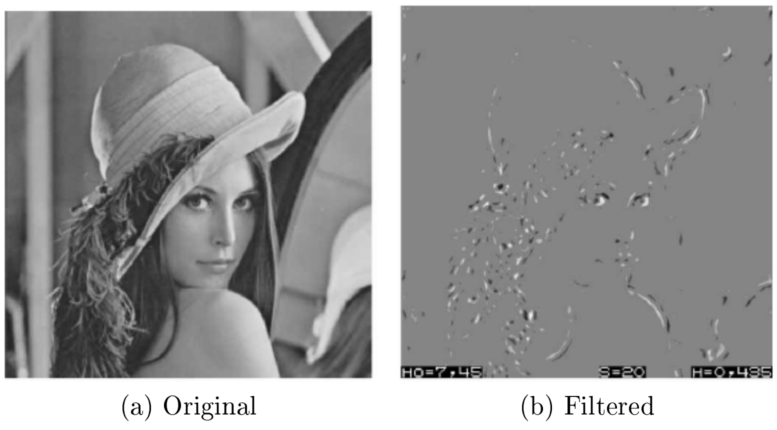


Figure 4.2: Application of i2D filter to Lena picture 4.2a leading to a representation with non-redundant features only 4.2b. (`source:` Zetzsche and Nuding (2005))

4.1.1.2 Sensorimotor features

As already stated above, images are represented as sensorimotor features. Such a feature consist of a triple of two foveal features (feature vectors at specific fixation location) and the relative position between them representing the saccade performed. This triple is abbreviated as FAF (Feature - Action -Feature). The foveal features are obtained in the preprocessing stage describe above. The relative position is encoded in a qualitative way by abstracting the numerical angle and distance (Schill et al., 2001).

4.1.1.3 Learning

During the learning stage, the system learns to associate the sensorimotor features with individual scenes. By executing all possible saccades, a statistic with relative occurrence frequencies is obtained for every picture. A specific FAF thus supports an individual scene up to a certain degree and is represented in the knowledge base as a hypothesis for an image and the corresponding value provided by a eye-movement. The results are stored in a Matrix $M = \{x_{ij}\}$ where each row is associated with a eye-movement and each column is associated with a hypothesis. The support $r_i(H_j)$ for a particular hypothesis H_j given by a particular eye-movement E_i is calculated by the number of times this eye-movement was performed on the particular image divided by the total occurrences of this eye-movement

$$r_i(H_j) = \frac{x_{ij}}{\sum_{k=0}^{j} x_{ik}} \tag{4.1}$$

The hypothesis space T is a strictly hierarchical one with every non-leaf object corresponding to a class and every leaf corresponding to an object[1]. Any element from T can be thus interpreted as a subset of the set of all objects θ (Schill et al., 2001). So far the actual hierarchy has been generated manually which should be changed by the output of this work.

The ratio $r_i(H_j)$ corresponds to a basic probability assignment (bpa) in the Dempster-Shafer *Theory of Evidence* framework (Dempster, 1967; Shafer, 1976).

4.1.1.4 Classification

The classification of images done by performing sequences of saccades, and gathering pieces of evidence for (a set of) hypothesis with every saccade.

The interesting point is, how the system determines which saccade should be performed next in every step. The strategy used for that matter is IBIG (Inference by information gain). The basic idea is to determine the piece of data

[1]An individual object can be of course interpreted as a class, too.

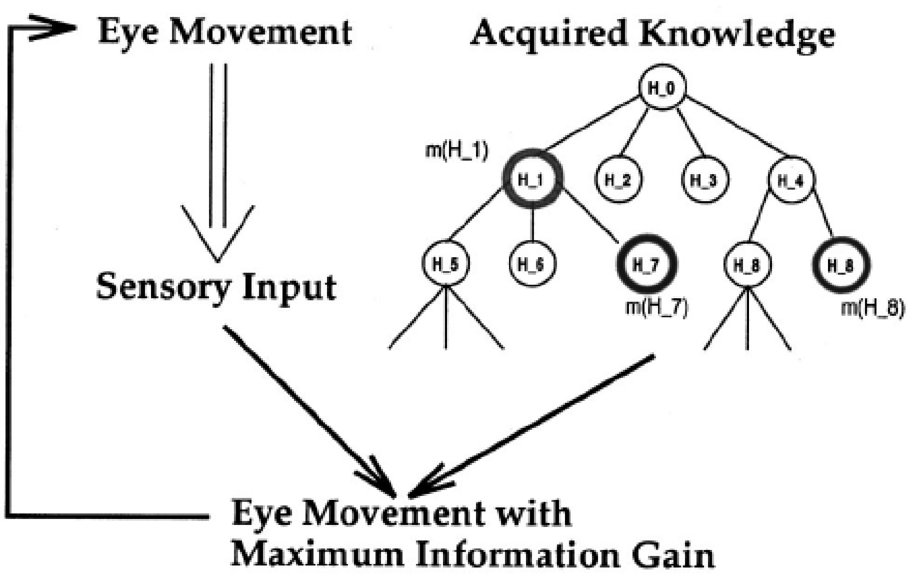


Figure 4.3: IBIG cycle. Given a sensory input and the corresponding belief distribution in the hierarchy, a saccade with expected maximum information gain is calculated. The resulting eye-movement leads to the next sensory input and closes the cycle. (`source:` Schill et al. (2001))

which holds the largest information gain with regard to the current evidence distribution of hypotheses space (Schill, 1997).

The information gain corresponds to the difference of the current belief distribution m_T and the potential belief distribution $\hat{m}_T$. The current belief distribution m_T is computed from all the evidence gathered so far. The potential belief distribution $\hat{m}_T$ is based on all evidence so far plus yet uncollected evidence.

$$I = |m_T - \hat{m}_T| \tag{4.2}$$

In the context of scene analysis, the evidence so far corresponds to the saccades executed so far. The potential evidence is thus a possible saccade which could be performed next. Possible saccades are all those with the current fixation point as the starting point and all not yet visited fixation points as a target. The belief is calculated by combination of the ratios of the performed sensorimotor features. After selecting a promising saccade, the belief distribution is recalculated and the selection process starts again. Figure 4.3 illustrates the cyclic process.

By selecting the sensorimotor features promising the highest information gain, the overall uncertainty with regard to the belief distribution of the hypotheses space is supposed to be minimized.

4.1.2 SMX and VAIN

The principle of using sensorimotor features for representation has been already extended from visual scenes to spatial representation.

SMX (Sensorimotor Explorer) (Wolter, 2006) introduced an VR-based agent system which was developed to solve navigation tasks (with the focus on localization) in a virtual environment. For the spatial representation, a sensorimotor approach very similar to the Okusys approach described above (Subsection 4.1.1, page 44) has been used. Here the sensorimotor features consist of fixed locations or views[2] which are connected by movement actions.

VAIN (Vision-based Active Belief Navigator) (Reineking, 2008) focused on extending the agent's capabilities in terms of localization. This included active localization (seeking for optimal disambiguating actions) as well as recovery from fatal localization situations (e.g. after kidnapping the agent).

4.2 Data description

The following section deals with description of the input data used for clustering. This covers basically the data used in *Okusys* (see section 4.1.1, page 44), i.e. the representation of scenes / images with sensorimotor features.

4.2.1 Representation of sensorimotor features

A sensorimotor feature like it is used in *Okusys* is a triple consisting of a starting feature, an action and a target feature. It is usually referred to as *FAF* (feature, action, feature) for clarification. In the case of images, the starting and target features are foveal features representing the image "content" at a specific image location. They correspond to fixation location on a scene. The action describes a relative movement from the starting to the target feature, it has thus a direction. Figure 4.4 illustrates a FAF. The explanation of the individual values is given below.

4.2.1.1 Stored values

A foveal feature contains the following values:

- Opening angle: angle defined by the two of an intrinsic i2D-feature. The edges forming the angle are determined by the outputs of linear orientation selective filters (Schill et al., 2001).

- Orientation: direction of angle opening with respect to image.

[2]An analogy to neural "place cells" discovered in the brains of mammals which fire when a specific location known to the subject is perceived

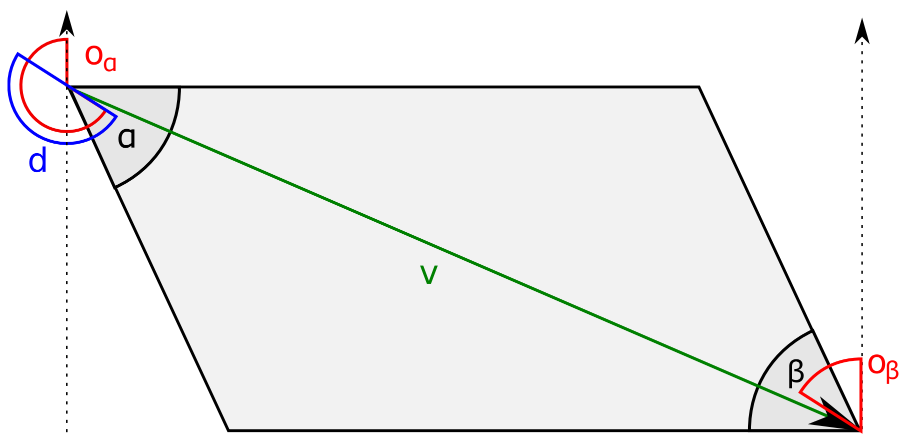


Figure 4.4: Sensorimotor feature in a parallelogram. α and β are the *opening angles* of two foveal features located in the corners. o_α and o_β depict the *orientations* of these angles and d shows the *difference angle* between them. v is the *directional vector* representing the saccade.

- Color: Color at specific position[3], represented by RGB values.

An action representation contains the following values.

- Distance: relative distance between starting and target feature.
- Difference angle: difference between the orientation of the opening angle of those features.
- Relation angle: angle between the two features in relation to the image.

The values making up foveal features and actions are obtained by numerical measurements, but stored as (pseudo-)qualitative values.

- Distance: The distance between starting and target features is calculated as the Euclidian distance between the pixel locations. The actually stored value is a mapping to (pseudo-)ordinal values ("very close", "near", "far" and "very far").
- Angular values: Those are obtained as numeric data, too, and mapped onto intervals with 10° or 5° width.

[3]The *Okusys* system as described in Schill et al. (2001) has actually no color support. It has been added recently.

This is insofar relevant as to decide whether numeric similarity measures can be applied to those data. The angles with its interval representation pose no problem. The (pseudo-)ordinal distances on the other hand have to be transformed into suitable numeric values which reflects the relative distance between individual values. Since the range covered by the original distance measures is known, such a transformation is applicable, for instance by taking the upper limit of the interval or the average distance as a numeric representation[4]

4.2.1.2 Comparing FAFs

Originally FAFs have been treated as individual symbols, e.g. a pair was either equal or not based on whether all values matched or not. They can be best regarded as *nominal* attributes when compared that way. There is neither a specific order nor a similarity expressed in numerical values. Still due to the nature of the values (or rather the initial measuring methods) it would be perfectly fine to impose a distance based dissimilarity measure.

4.2.2 Scene representation

As already said a scene is represented as a set of sensorimotor features. Figure 4.5 shows a very simple scene: original image and filtered one with sensorimotor features.

The entire set can be regarded as a fully connected graph with the foveal features corresponding graph nodes. A sensorimotor feature represent an edge[5]. As the action is of such a feature is directional, the representation requires two sensorimotor features for every pair of foveal features.

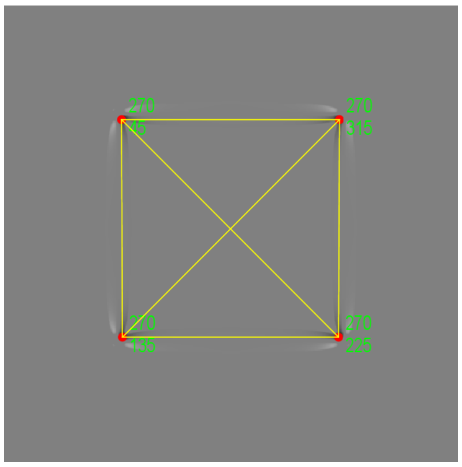
One can already guess that for a relatively low number of foveal features (and thus fixation locations) the number of total sensorimotor features becomes pretty high. The number of edges of an undirected fully connected graph corresponds to $|E| = \frac{|V| \cdot (|V|-1)}{2}$ with $|V|$ being the cardinality of vernices and $|E|$ the cardinality of edges. Since each undirected edge is represented as an directed one in this case, the cardinality increases to $|E| = |V| \cdot (|V| - 1)$. Figure 4.6 illustrates the extracted sensorimotor features for a real-world image. It has 44 extracted foveal features which sum up to 1892 sensorimotor features.

The set of sensorimotor features is unordered. There is no particular inherent starting point for looking at an image in general, even though one could possibly define an initial fixation location in some way.

[4]Note that it is not meant to calculate the original distance. It is just a scale transformation in order to reflect the original differences in distance.

[5]Actually only the action (saccade) should be regarded as an edge since it is the connecting element between individual nodes.

(a) Original



(b) Filtered

Figure 4.5: Image showing a black rectangle on a white background. Unfiltered (a) and filtered (b). The corners of the rectangle are the foveal features (red dots) found in this scene; the numbers state the opening angle (top) and orientation (botton) of each foveal feature. The yellow lines show the actions (saccades) which can be performed. Note that for each line, two actions are possible since the actions are directional.

4.3 Approaching the problem

4.3.1 Tackling the similarity of scenes

The question at hand is, how can a value expressing the similarity of individual scenes be obtained? There are two very basic way to tackle this problem:

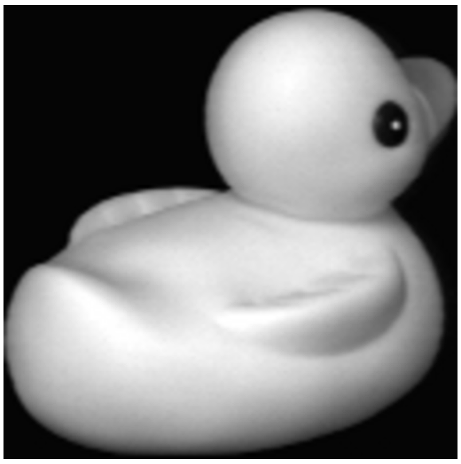
- Designing a similarity measurement specifically suitable for representation at hand
- Transforming the current representation into another one which is suitable for application of well-known similarity measures

4.3.2 Sequences similarity

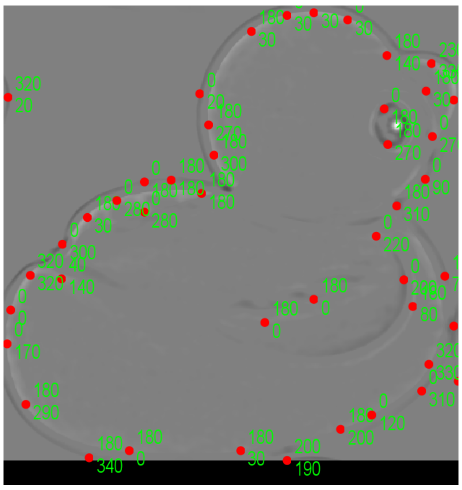
To get a notion of how similarity might be measured, one can take a closer look at the way Okusys (section 4.1.1, page 44) analyzes scenes. Its behavior might give some clues.

Basically Okusys takes a sensorimotor feature[6] and looks up the support of this FAF for a particular scene (or a set of scenes). At this point a particular sensorimotor feature might support one or more scenes depending upon

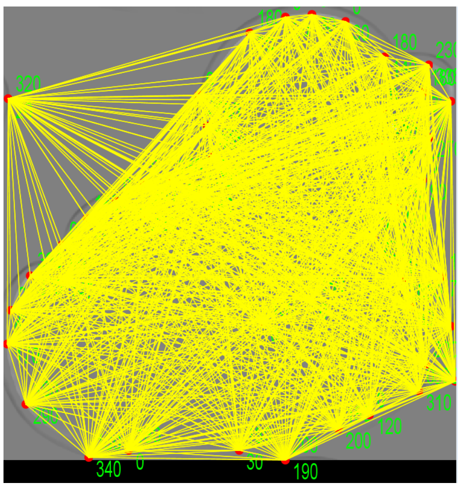
[6]Thus it performs a saccade from one fixation location to another in biological sense



(a) Unfiltered



(b) Filtered without saccades



(c) Filtered with saccades

Figure 4.6: Three views of a plastic duck: (a) unfiltered, (b) after filtering showing the foveal features only with opening angle (top number) and orientation values (bottom number) and (c) finally filtered showing foveal features and saccades. (`image source:` part of COIL-20 image library (Nene et al., 1996))

whether it has been seen in one or more scenes during the learning stage. Ambiguity arises if a particular FAF supports more than one scene obviously. Since sensorimotor features are processed sequentially with the individual supports being combined, there can be only ambiguity if entire sequences are found in more than one scene.

Comparing possible sequences of sensorimotor features of pairs of scenes for equality and taking this as a measure for similarity would certainly work. But one can already suspect that such an approach will pose problems in terms of computability. For a precise measure, actually all possible and thus unique sequences must be generated which after all means to permutate. For a given number of foveal features i, the number of unique full-length sequences is $(i-1)!$ which puts the generation of sequences into the complexity class of $O(n!)$. This complexity already rules out the generation of complete sequences; with regard to scenes with 40+ foveal features, it would require a life-time to obtain the sequences for a single scene alone.

Limiting the length of a sequences would reduce the complexity and might serve as a broad approximation. Still, even sequences with length 2 or 3 already explode with regard to absolute numbers. A scene with 44 foveal features generates 79,464 2-step and 3,258,024 3-step sequences.

All in all, the idea of comparison of sequences is a rather bad one.

4.3.3 Transformation

The idea is to transform the unordered sensorimotor feature set into a representation which allows the direct application of numerical similarity measures. The target is to represent a scene as a vector of either integer or continuous values. Furthermore the dimensionality of features of a scene can be reduced.

The key to a numerical representation is to abstract from the individual FAFs by grouping sensorimotor features according to a similarity. The scene representation then consists of an occurrence count of abstracted sensorimotor features.

4.3.3.1 Clustering of sensorimotor features

The method applied here is to cluster the sensorimotor features with the help of a Self-Organizing Map. That way similar sensorimotor features are mapped onto topologically related nodes of the map and thus preserving the topology of FAFs in input space.

Degree of abstraction A specific node of the map represents a group of similar sensorimotor features in input space. How large this group is depends on the size of the map in the first place. It is obvious that a 1-node SOM would map all FAFs onto one group as the one extreme while a SOM with as many nodes as there are distinct FAFs would have a fair chance to group each sensorimotor feature into singleton groups. Still the latter case is not guaranteed; it depends too much on the actual convergence of the SOM. If two input features are only slightly different (i.e. one variable is different), they might be mapped onto one node in one case and onto two nodes in another depending on the organization of the local neighborhood of nodes in that area. During the learning process the density of input vectors plays a role as well. If there are many similar input vectors (many relative to the total number of input features) the SOM learns to distinguish those vectors "better" since the are presented more often and thus arrange the very same local group of map nodes in a rather fine way which as a result is sensitive to minor differences. On the other hand, another group of equally similar vectors in terms of absolute measures but fewer in numbers (making up a smaller proportion of the total population) would not be separated as fine. Basically if there are as many nodes as distance input vectors, it cannot be guaranteed that every distinct feature from input space is mapped onto distinct nodes in output space. Still this could be probably achieved by increasing the map size even further. The question is what size is necessary which is hard to answer. Still computation time will puts an limit to the number of nodes, since the convergence process will become very slow eventually.

Map parameters The distance measure used to compute the dissimilarity between an input vector and the weight vector of the node will be Euclidian metric. This metric is probably the most suitable for this task since it the absolute differences of values into account. A directional measure like the cosine would treat FAFs with equal relative distribution of values as equal which is not intended here. For instance, a sensorimotor feature with two foveal features having opening angles of 90° should not be equal to a FAF with opening angles of 180° (assuming the other values are equal for both). But this would actually be the case with a direction centered proximity measure.

The sensorimotor features need to be preprocessed first in order to be applicable. They need to be normalized to a uniform range, otherwise the Euclidian distance measure would prefer larger values. The attributes naturally range from 0 to 360 in case of the angles, 0 to 255 for the color values and 1 to 4 for distance. Without normalization the distance attribute would have close to no impact on the actual distance computed. The normalization will map all attributes to the range $[0 \ldots 1]$. Another aspect is that the color is expressed by three values[7] which means that color would have a larger impact on the distance than a single angle represented by one value. In order to neutralize this, the distances between color attributes will be weighted with a third only.

The grid structure of the map is a hexagonal grid. This layout should be gen-

[7]The three RGB channels

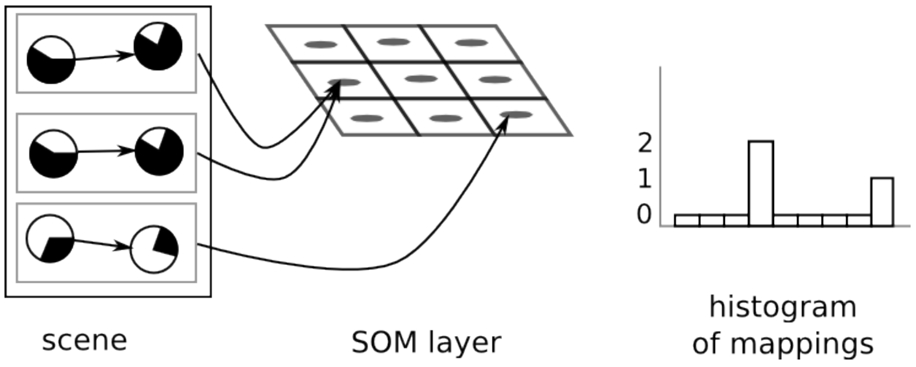


Figure 4.7: From FAF representation of scene to histogram. The FAFs representing a scene are mapped to a previously trained SOM. A histogram representation is obtained by counting those mappings.

erally preferred over the rectangular grid since it allows to place more neighbor nodes with the same distance around a node and thus gives more freedom to arrange a group of nodes.

The number of nodes will vary. Depending on the complexity of the scenes, there is a number of nodes above which the results do not change anymore. The problem is that this threshold cannot be estimated well; it is rather a question of experience. High number of nodes is favorable in order to achieve a low level of abstraction but not really necessary.

Other parameters: learning rate and number of repetitions: try and error

randomized node weight vectors. individual convergence process produce different results. local topology will be fine but global can be distorted. individual mappings of input vectors will vary in quality. somewhat like k-means: enough runs produce similar and good results but not all do. degree of distortion.

4.3.3.2 Numerical representation of scenes

The numerical scene representation is similar to the vector space model. While in the classical case of the application to document clustering, the vector components stand for individual terms found in documents, here the components are related to node of the Self-Organizing Map obtained previously. Whether the 2-dimensional structure of the output map is maintained or transformed into a 1-dimensional vector where each vector component is associated with a particular map node from the SOM does not really matter that much.

For all scenes the sensorimotor features are mapped onto the SOM. The occurrences of individual mappings (which node has a feature been mapped onto) are counted. Thus a histogram of SOM-mappings is obtained for each scene. Figure 4.7 illustrates this process.

The idea is that similar scene will trigger similar mappings since those scene are supposed to contain similar sensorimotor features. Entirely equal scenes should obviously produce exactly the same mappings. Similar scenes (similar in the sense that they share a common subset of sensorimotor features) on the other hand should only produce partially equal mappings.

This implies that histograms of these mappings are also similar. Regarding the histogram as a vector, the implication is that vector components share similar numerical values. This can be either in absolute terms (several components have exactly the same values or differ only slightly) or the relative distribution is similar. Ideally a specific subset of components would be collinear.

For counting the mappings, there are some possible approaches.

Winner-only counting The most simple case is to count the winner nodes only. For every mapping onto a specific SOM node, the occurrence is increased by one. This way does not take into account how far the distance / similarity between input vector (the sensorimotor feature) and the internal weight vector of the node was.

Distance integration This method takes the distance between input vector and the nodes' weight vectors into account. One can either integrate the winner node's distance only or integrate the distances to all map nodes as well. The latter method is similar to principles used in fuzzy clustering (i.e. fuzzy c-means Dunn, 1973; Bezdek, 1981) where each sample gets a assigned a class membership to all classes with the degree of membership depending on the similarity between sample and class (representation).

Winner neighborhood counting A last way to count would be to not only count the winner nodes mappings but to assign a partial hit to its neighborhood nodes, too. This method This can be achieved by applying a certain value (lower than the hit value for the winner) to nodes within a range according to the final neighborhood size[8]. The value applied can be calculated by the final neighborhood function as well.

There is actually no real reason why a certain neighborhood size or function should be preferred over another. One could just as well use any shape and size covering a limited number of nodes only. It is rather an intuitive decision to re-use the same function which shaped the neighborhood of nodes in the convergence process.

Figure 4.8 illustrates the cases described above.

The first case with counting the winners only should be actually sufficient for the sensorimotor features at hand. Since the values of angles and distance are already mapped onto intervals, small variances in the initial measurements are already ruled out by the interval representation. This leaves the case that sensorimotor features with different yet similar *intervals* are mapped to different

[8]Final size means the size of the neighborhood after finishing the learning process.

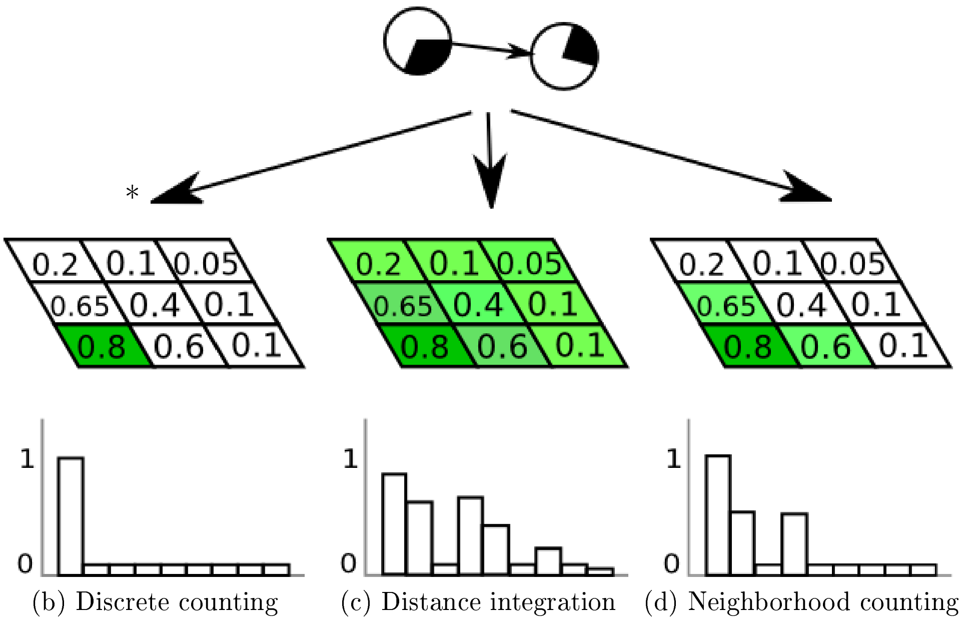


Figure 4.8: Counting the mappings in various ways. (b) shows the case of discrete counting of the winner node, (c) the integration of all distances and (d) the integration of the neighborhood nodes of the winning unit. The neighborhood function showed here is a simple cone function with size 2.

nodes which might produce histograms that do not match at all in the extreme case. An example might be the comparison of the features of a rectangle with those of a parallelogram having angles only slightly below/above 90° so that the mappings are also slightly off. From an Okusys perspective, it might be actually fine, depending on whether the parameters used for comparison would treat these sensorimotor features as equal or not. If they are not equal, then the mis-mappings are plainly irrelevant.

In order to overcome these problems, the other two methods incorporate more nodes than the winner only. The quality of distance integration over the entire map is depending a lot on the distances calculated for the node which are way off. In the case of larger maps with some hundred nodes, the distances to the non-matching nodes can sum up to a rather large value, simply due to the number of nodes which makes the results very vague in the end. Therefore a limitation of the nodes which should be Incorporated is recommended. Limiting it to the local neighborhood of a node looks like a good compromise since the local neighborhood of node in the output layer is supposed to carry a topological relation to the winner node.

Still this approach is probably most useful for real, ratio values since there slight differences might lead to mis-mappings more often then for interval values, which already abstract from the measured ratio values.

4.3.3.3 Interpretation

How can one interpret this approach, especially from a cognitive point of view?

If one accepts the self-organization map as a modell which reflects the biological process of mapping sensory input, then the learning and mapping of sensory input can be regarded as cognitive. Still, this usually just holds for the mapping of sensory information from one source, i.e. the visual system. With regard to this work, it would mean the mapping of foveal features only which represent visually perceived features. Furthermore it holds also true for motor actions like a saccades (see (Roucoux et al., 1980; Wurtz and Albano, 1980) for experiments on gaze directions). What can be disputed is the application to a triple of representing visual and motor actions in one structure and the mapping of it.

With regard to the Self-Organizing Map as an ANN, the mapping of FAFs can be seen as neuron(s) firing when presented a certain sensory information. Extended to the iterative mapping of the entire set of FAFs representing a scene, one can regard it as which neurons fired how often when a particular scene was the input.

4.4 Hierarchical clustering

There are two approaches used to obtain the actual hierarchy. The first one is the classical agglomerative way of dendrogram building. In a bottom-up singleton clusters containing the scenes will be merged. This will be done with several distance measures as well as linkage rules.

The second approach is a splitting method. In a top-down process the sets are divided into smaller sets by clustering them with a SOM iteratively.

4.4.1 Hierarchy

The hierarchy which should be finally generated, is supposed to structure the original patterns in a meaningfull way. What is meaningful with regard to patterns which represent a scene by a set of sensorimotor features? The assumption is that objects should be grouped together according to some features they share[9] which is in this case a subset of sensorimotor features shared by a certain set of patterns.

The non-leaf nodes in the hierarchy represent *classes* while the leafs are the actual objects. The classes will finally contain the set of features common to all sub-classes and leaves. This set is basically the intersection of all sets from that specific subtree, so that a particular sensorimotor feature associated with

[9]Again, there is not objective preference for a relation of objects over another according to the ugly duckling theorem (see 2.5.1, page 31). Extending this idea to a hierarchy, it implies that any hierarchy would be as good as another.

a class is part of all leaves that are children. From the Okusys perspective, such a FAF associated with a class supports this class and, if broken down to individual objects, all of the objects in the tree of which the class is the root.

Intuitively the higher a class is in the hierarchy, the lower is the number of common features. Or at least, it cannot increase from a class located lower to a higher one. The root of the hierarchy thus contains features only which are common to all objects. As an interpretation, "seen" such a root feature does not hold any evidence for a specific scene or object on its own.

4.4.2 Agglomerative clustering

The first approach is to apply the agglomerative clustering methods described in section 2.4.1, page 26. The patterns used as an input are the histograms obtained from mapping the sensorimotor features to a self-organizing map.

4.4.2.1 Similarity measures

As described in section 2.2, there is a number of similarity measures for patterns which are represented by numerical values.

Euclidian, cosine and Tanimoto measure will be used here in order to compute the similarity of patterns. Considering the assumption that patterns are similar if they share common features and furthermore that even though the absolute numbers of features are not equal, but the relative distribution is, one can assume that cosine and maybe Tanimoto measures are the most suitable for this problem.

The question is whether the vector representation of the patterns reflects the (symbol-like) sensorimotor set representation well enough for the similarity measure to compute a useful value.

4.4.2.2 Linkage rules

With regard to the available linkage rules, the impact on the quality of the results depends mainly on the nature of the input data. It is rather hard to assess which rule might be the best before applying it to the data. From a general perspective, the average linkage rules should produce the best results, especially Ward's linkage rule.

4.4.2.3 Output

The output generated here is a dendrogram of course. Being a binary tree, it is not necessarily the best hierarchy type. The class hierarchy in general actually allows more then two subclasses or objects directly attached which makes a binary hierarchy a sub-optimal representation. Still, a binary hierarchy can just express the very same information, as showed in figure 4.9. But a hierarchy where three subclasses are placed directly under a common parent naturally implies that the subclasses share nothing but whatever is represented in the parent node. In the case of sensorimotor feature sets, the set associated with the parent is the intersection of all children and furthermore there is not combination of two children with a higher cardinality of the intersection. In the case of a binary tree, this is not obvious. Just to stick with three nodes as an example, here two node must be grouped together and the third one will be added on top of them. Without any further information, it is not clear whether these two grouped first share more features or not. Nevertheless, only the top class node would hold support for the leaves beneath it, so that the actual information given by this representation would not be any different.

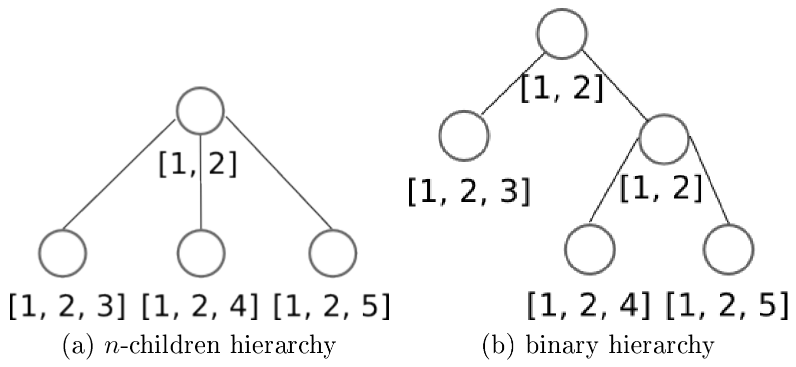


Figure 4.9: (a) General and (b) binary hierarchy. Note that the top-most node in each hierarchy contains the same subset of features (denoted as figures here) and thus represent the leafs of the trees equally well.

4.4.2.4 Assessment of dendrogram

The assessment of the results will be conducted in the following ways.

Cophenetic correlation The cophenetic correlation will be calculated for a generated dendrogram. It allows to judge on the quality of linkage rules used to build the hierarchy. Unfortunately it should not be used to compare hierarchies produced with different similarity measures.

f-measure In the case where class labels are available, the f-measure will be used for evaluation as well. It allows for the comparison linkage and similarity measures across all combinations.

CHAPTER 5

Application and Tests

The following chapter deals with testing the clustering algorithm with artificial images, and applying it to real world images of objects.

The test images were designed with a specific underlying hierarchy in mind.

The application to real world images was performed on images taken from the COIL-20 image library (Nene et al., 1996).

The outputs presented here are exemplary. In appendix C, page 83 provides all combinations of linkage and distance variations for some of the test sets.

Please note that the results presented here can be regarded as exemplary certain sets, artificial as well as real world ones.

5.1 General remarks

One very basic problem which arises while testing the algorithm is related to the input data generated by Okusys. It is totally unpredictable which features Okusys might extract from an image if applied to real world scenes. Even if two images show the very same object, but the images are taken from slightly different perspectives, one cannot be sure that the same features will be extracted, especially with regard to position and orientation in the image. Of course, the entire sets will not be totally different; otherwise it would pose a major obstacle to measuring the similarity between scenes. Still in order to rule out the “Okusys factor”, i.e. any distortions caused by differences in extracted foveal features, the testing will be performed on synthetic images where the feature extraction will be rather predictable.

The image sets generated for this reason contain geometric objects with clear edges and corners, and are furthermore restricted to black and white colors. The feature extraction on such images should produce rather deterministic results with regard to which foveal features are chosen from the objects found in the scenes.

Even with such primitive images, the actual extracted features may vary. This is especially the case with round edges. Here any position on a curved edge can be regarded as a *i2D*-feature. Furthermore anti-aliasing effects like stair steps can distort opening angle and orientation of the foveal features.

5.2 Repeated patterns

The images used in this tests contain objects which are repeated in varying numbers across the images. For instance figure 5.1 shows a test set with different numbers of rectangles, ranging from one to four objects.

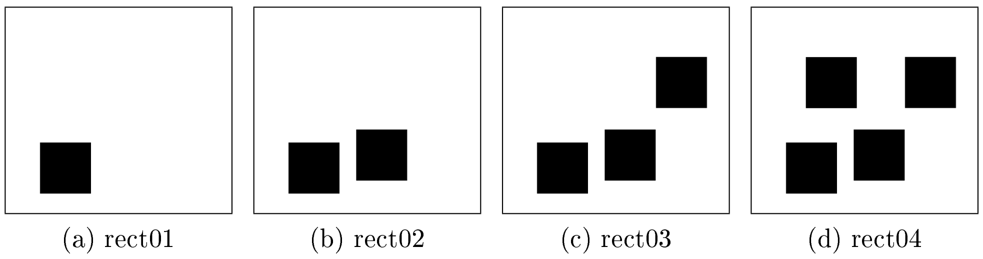


Figure 5.1: Test set with different numbers of rectangles. The number increases by one from image to image.

The goal is to see whether these images can be ordered according to the subsets of features they contain. Obviously 5.1c and 5.1d share the largest common subset of features since the entire image of 5.1c is part of 5.1d. Intuitively these two scenes should be placed at the bottom of the hierarchy and thus be grouped first. Accordingly 5.1b and finally 5.1a should be added to the hierarchy.

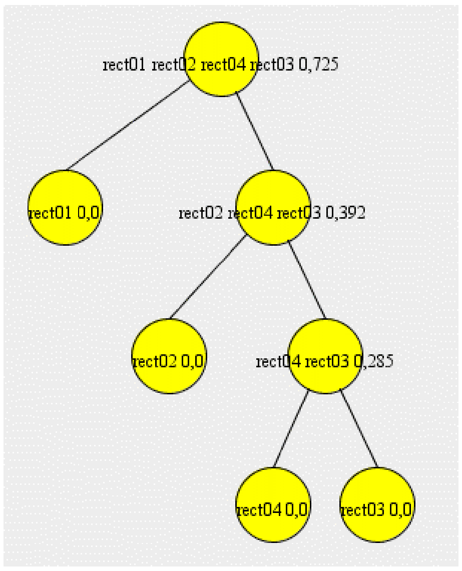
Figure 5.2 shows two hierarchies generated with single linkage[1]. Cosine measure was used in the case of 5.2a and Euclidian for 5.2b, resulting in different hierarchies. It is not surprising that the cosine measure captures the nature of the data in the supposed way since it regards similar distributions of values as similar; in contrast the Euclidian measure does not but only regards similar absolute values as “close”. As the number of rectangles increases, the absolute number of features generated by this structure increases as well while the relative distribution remains the same for the features “inside” the boundaries of the rectangle. Furthermore the images with more rectangles also produce more sensorimotor features connecting the rectangles which basically means there the absolute number of similar features is higher.

[1]Appendix C.1 provides all combinations.

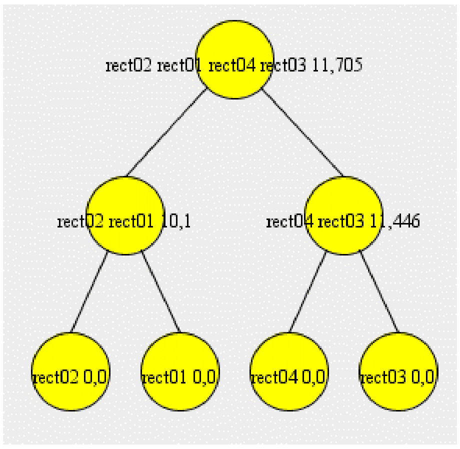
The linkage has no particular impact for these data; the resulting hierarchies are exactly the same for each distance measure. Cosine and Tanimoto produce the same results, too. This agrees with other observations that cosine and Tanimoto behave similarly for data with similar distribution of values.

The cophenetic correlation is pointless in this case since all hierarchies generated with a particular proximity measure are identical. As for the f-measure, it would require labeled data and I doubt that there is a way to separate these images into meaningfull classes.

The test set 02 and 03 (appendix C.2 and C.3) feature the same structural layout, only with different geometric figures. The results do not vary from those discussed above.



(a) Cosine dissimilarity



(b) Euclidian dissimilarity

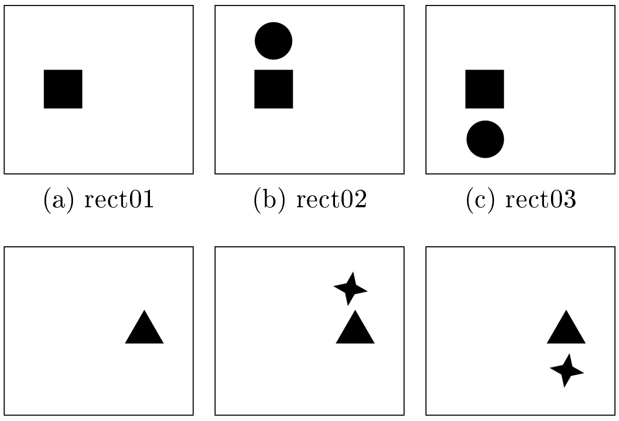
Figure 5.2: Hierarchies generated based on set01 with single linkage. Cosine measure (a) and Euclidian measure (b) serve as dissimilarity measures.

5.3 Multiple subsets

Figure 5.3 shows a set of synthetic images which can be divided into two classes, each class containing three images. These classes have either a rectangle or a triangle as a central element.

The expected outcome of the clustering should be a separation of these two classes, so that each class resides in its own subtree.

Subjective assessment Figure 5.4 shows the outputs of Ward's linkage with Tanimoto, cosine and Euclidian proximity measure. One can see that the hier-

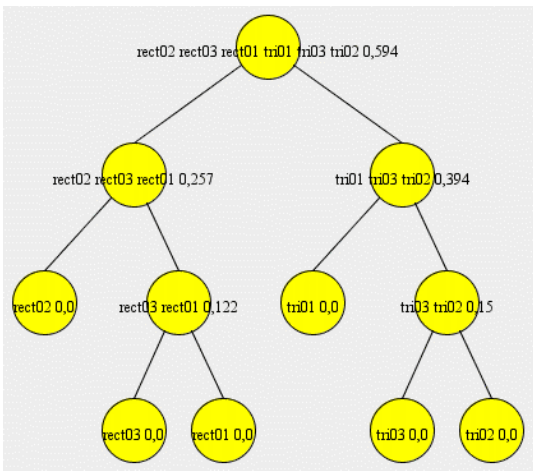


(a) rect01 (b) rect02 (c) rect03

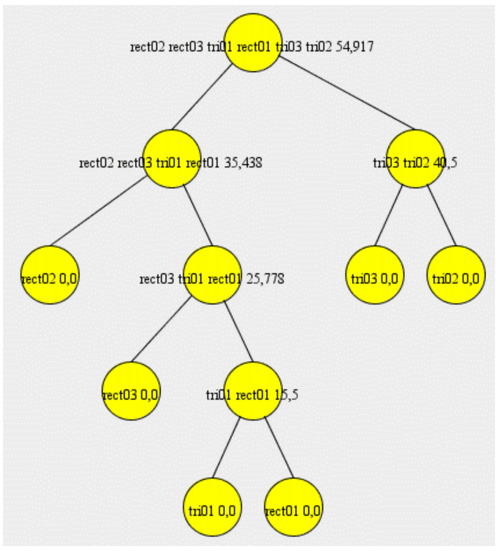
(d) tri01 (e) tri02 (f) tri03

Figure 5.3: Test set 06

archies using Tanimoto and cosine measure group the objects according to the underlying classes while the one with Euclidian measure does not. Taking a look on the other linkage rules (all outputs are shown in figures C.22 to C.24), it is clear that the linkage rule does not have any impact; the quality is entirely depending on the proximity measure.



(a) Tanimoto & Cosine


rect02 rect03 tri01 rect01 tri03 tri02 54,917
rect02 rect03 tri01 rect01 35,438
tri03 tri02 40,5
rect02 0,0
rect03 tri01 rect01 25,778
tri03 0,0
tri02 0,0
rect03 0,0
tri01 rect01 15,5
tri01 0,0
rect01 0,0

(b) Euclidian

Figure 5.4: Hierarchies for set 06 with Ward's linkage and (a) Tanimoto (and Cosine) as well as (b) Euclidian. Note that Tanimoto and cosine do not produce exactly the same values but the structure of the hierarchy is the same.

Cophenetic correlation and f-measure Table 5.1 shows the cophenetic correlation as well as the f-measure for all combinations of linkage rule and proximity measure.

The f-measure supports the statement from above that the Tanimoto and cosine proximity measures produce hierarchies which reflect the class sets while the Euclidian does not. The f-measure for the two first-mentioned measures is 1.0 for all linkage rules. For Euclidian it is lower than 1.0, scoring the highest for complete and Ward's linkage. The good score of the complete linkage implies a sphere-like, compact but not well-separated structure of the data.

Table 5.1: Cophenetic correlation and f-measure for test set 06 with (a) cosine, (b) Tanimoto and (c) Euclidian.

Linkage	**CPCC**	**f-measure**
Single	0.976	1.000
Complete	0.974	1.000
Average between groups	0.985	1.000
Average within group	0.982	1.000
Ward	0.960	1.000

(a) cosine

Linkage	**CPCC**	**f-measure**
Single	0.967	1.000
Complete	0.964	1.000
Average between groups	0.978	1.000
Average within group	0.976	1.000
Ward	0.969	1.000

(b) Tanimoto

Linkage	**CPCC**	**f-measure**
Single	0.683	0.708
Complete	0.756	0.829
Average between groups	0.790	0.762
Average within group	0.786	0.762
Ward	0.738	0.829

(c) Euclidian

5.4 Application to real world object images

Figure 5.5 shows a set of images of real world objects. This is a very limited subset of images from the COIL-20 image library (Nene et al., 1996) which contains pictures of 20 objects taken from different, fixed perspectives. The entire library consists of 1,440 images which unfortunately is way too much to be used here. My attempts to apply the clustering to the entire set failed at the stage of extraction of sensorimotor features by Okusys due to memory limitations.

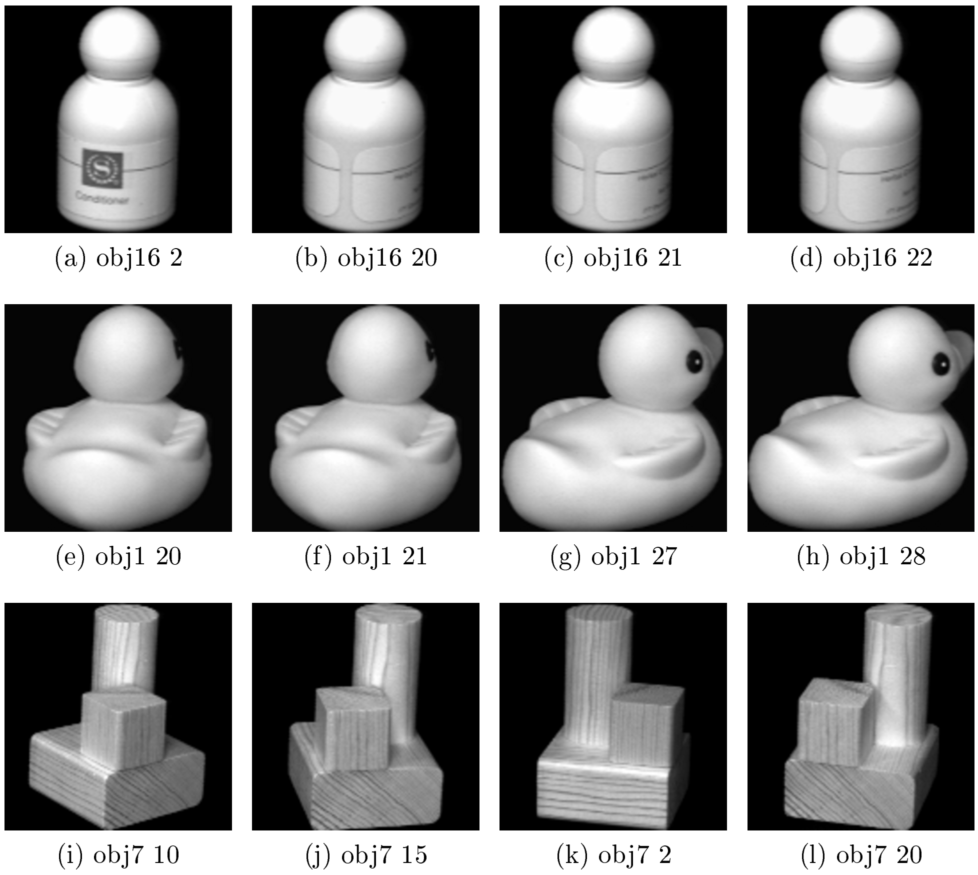


Figure 5.5: Test set Coil

Parameters This test was performed with a SOM with 60 × 40 nodes, 0.3 initial learning rate and 15 iterations of each pattern (sensorimotor feature).

Data description The set chosen depicts three objects, each from four different perspectives. The similarity between these perspectives is present to a certain extend, even though an object like the duck does certainly not look alike from behind and from the side.

As these objects are labeled, thus there is a known class for each, the class labels can be used for assessment of the hierarchy. Ideally all images belonging to an object should be placed in a separate subtree before being merged with the other subtrees.

Subjective assessment From the outputs (see section C.8, page 112 for all outputs) one can say that the hierarchies using Euclidian distance (figure C.32, page 118) as a proximity measure are the worst overall. They mix `obj1` and `obj16` (the duck and the pill tube) on a low level and add `obj7` (wood toy) one by one later.

As for Tanimoto and cosine as similarity measures, they perform reasonably better, even though not with all linkage rules. The single and complete linkage, there are severe distortions (see figures C.30a, C.30b, C.31a and C.31b, p. 113–116).

The average linkage rules seem to have less to separate the objects according to the classes even though only Ward's linkage the hierarchy according to the assumed class structure correctly for both, Tanimoto and cosine measure.

Cophenetic correlation and f-measure Table 5.2 shows the cophenetic correction and f-measure for the COIL images. The f-measure can be calculated for this set since the class assignment of each image is known.

Comparing the outputs of the cophenetic correlation with the subjectively judged hierarchies, one can easily say that there is a discrepancy. This measure produces the highest values for *average linkage between groups* which is not the best. Ward's linkage on the other hand scores considerably lower, even though intuitively it performs the best. Most striking is probably the huge value for all Euclidian based hierarchies which are all close to 1.0.

The f-measure which aims at expressing how well the hierarchy reflects the class structure is more usefull as an assessment measure. It reveals what has been already stated: Ward's linkage obtaining a 1.0 score for Tanimoto (see figure 5.6) and cosine proximity performs the best by producing a particular cluster for each class. The f-measure for all Euclidian based hierarchies lower than the worst linkage with Tanimoto and cosine which states the inappropriateness of this proximity measure for this set.

5.5 Influence of SOM size

From experimental experience with this system, I can say that the impact of the map size is present to a certain degree. Still it highly depends on the actual data. In the case of the test sets consisting of artificial images at the beginning

Table 5.2: Cophenetic correlation and f-measure for COIL set with (a) cosine, (b) Tanimoto and (c) Euclidian.

Linkage	**CPCC**	**f-measure**
Single	0.575	0.794
Complete	0.737	0.905
Average between groups	0.760	0.915
Average within group	0.722	0.915
Ward	0.657	1.000

(a) cosine

Linkage	**CPCC**	**f-measure**
Single	0.707	0.861
Complete	0.830	0.915
Average between groups	0.854	1.000
Average within group	0.830	0.915
Ward	0.786	1.000

(b) Tanimoto

Linkage	**CPCC**	**f-measure**
Single	0.988	0.656
Complete	0.986	0.675
Average between groups	0.990	0.656
Average within group	0.980	0.656
Ward	0.969	0.675

(c) Euclidian

of this chapter, there is close to no influence. The mappings are alredy very sharp, so that only a very small map (say 2x2 size) would have any impact.

With regard to the real world images, the impact is bigger as one can imagine. Rather stable results were obtained with maps consisting of 600+ nodes. Below the hierarchical output varied from run to run significantly. Still even with 600-1000 nodes, the hierarchy not alway perfect (f-measure 1.0 when evaluated with class labels). From 1500 node on, the results did not change anymore, so this value can be seen as an uppper threshold above which the quality does not vary significantly anymore.

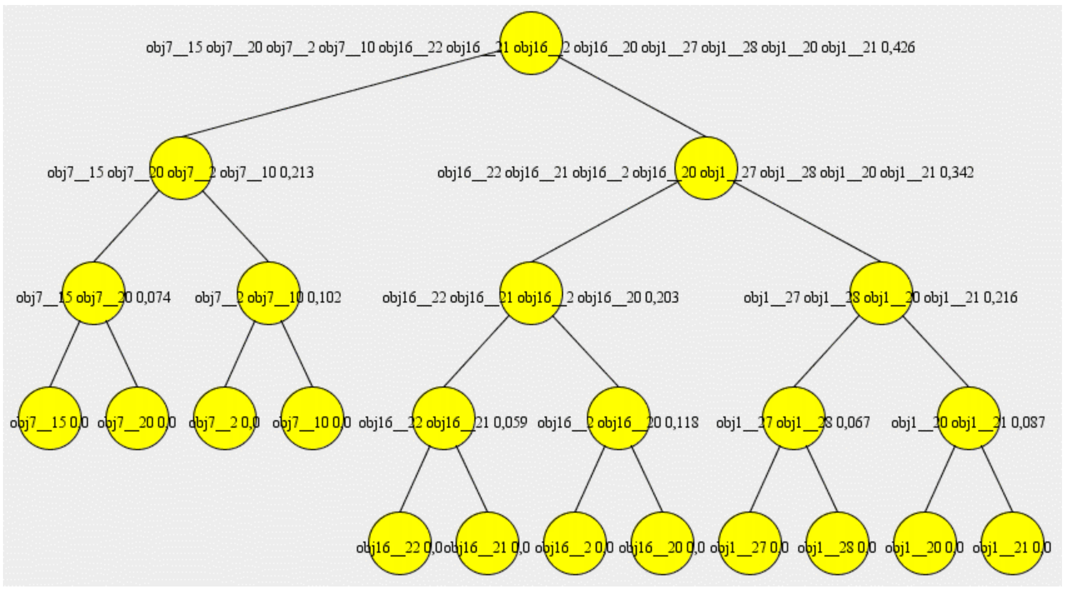

obj7__15 obj7__20 obj7__2 obj7__10 obj16__22 obj16__21 obj16__2 obj16__20 obj1__27 obj1__28 obj1__20 obj1__21 0,426
obj7__15 obj7__20 obj7__2 obj7__10 0,213
obj16__22 obj16__21 obj16__2 obj16__20 obj1__27 obj1__28 obj1__20 obj1__21 0,342
obj7__15 obj7__20 0,074
obj7__2 obj7__10 0,102
obj16__22 obj16__21 obj16__2 obj16__20 0,203
obj1__27 obj1__28 obj1__20 obj1__21 0,216
obj7__15 0,0
obj7__20 0,0
obj7__2 0,0
obj7__10 0,0
obj16__22 obj16__21 0,059
obj16__2 obj16__20 0,118
obj1__27 obj1__28 0,067
obj1__20 obj1__21 0,087
obj16__22 0,0
obj16__21 0,0
obj16__2 0,0
obj16__20 0,0
obj1__27 0,0
obj1__28 0,0
obj1__20 0,0
obj1__21 0,0

Figure 5.6: Hierarchy with Tanimoto measure and Ward's linkage

CHAPTER 6

Existing applications

6.1 Saccadic scene representation

Beside the work already introduced before, there are not too many which deal with scene representation / recognition based on saccadic movement.

The idea of using saccadic movements for recognition originates in the research of Noton and Stark (1971) who introduced the concept of scanpaths. Their finding was that test subjects reproduced sequences of saccades when recognizing patterns they have learned before.

Salah et al. (2002) use a system for handwritten digit as well as face recognition. It consists of attentive selection of fixation spots which are analyzed by a neural network for the content of the fovea. Finally on the associative level, the saccades are integrated over time by an observable Markov Model.

Hacisalihzade et al. (1992) investigate deterministic and probabilistic approaches to scanpath comparison. The deterministic approach features sequences of saccades which are compared with string similarity measures. The probabilistic part models the sequences as a Markov Model.

Rybak et al. (1998, 2005) developed a model of attention-guided visual perception and recognition. The recognition is done by a behavioral program using learned saccades (motor action memory) and predicted image fragments input (sensory memory) for each saccade.

Kikuchi and Fukushima (2000) abandon the scanpath approach and store sensorimotor features as sets. Recognition is done by comparing the currently perceived with learned sets. The model is build with neural networks.

6.2 Clustering

The number of existing applications in the field of clustering is legion. A complete overview is impossible anyway, but even a thoroughfull collection would fill an entire textbook. I will try to stick with hierarchical clustering applications since they are the most relevant with regard to this work.

Important general purpose hierarchical clustering approaches have been already covered in section 2.4.2, page 29.

6.2.1 SOM hierarchical clustering

The idea of using hierarchical SOMs probably originates from the work of Miikkulainen (1990). The nodes of a SOM are connected to further SOMs so that a hierarchical structure of stacked maps is generated. Each layer can be regarded as a preprocessor for any further layers.

Herrero and Dopazo (2002) and Wang et al. (2002) use a SOM and hierarchical clustering to obtain a hierarchy of gene expression patterns. First DNA microarray patterns are clustered with a SOM to reduce the dimensionality of the patterns. The input patterns are generalized by the map nodes' prototypes. Afterwards agglomerative hierarchical clustering is done on the SOM map nodes. (Yacoub et al., 2001) use this method to classification of ocean color. Furthermore they suggest the squared error as a threshold criterion to obtain a flat list of clusters from the hierarchy generated.

Dittenbach et al. (2001) proposed the GHSOM, the growing hierarchical SOM, which stacks multiple layers of SOMs. The quantization error, squared distances between node prototype and mapped patterns, compared to a threshold indicates whether a node has to be split any further.

6.2.2 Image clustering

Krishnamachari and Abdel-Mottaleb developed a system for image retrieval for query search. The images are indexed with clustering methods for faster retrieval. The similarity measure used here is based on color histograms.

Brunner (2006); Brunner and Burkhardt (2005) uses hierarchical clustering for grouping edges extracted from an image in structural image analysis. Keuchel et al. (2004) use a hierarchical partition tree for segmentation of images.

6.2.3 Symbol clustering

In many applications, the input data consist of symbolic data. Some approaches which deal with hierarchical clustering are introduced next. Maybe the most prominent domain where this problem arises is document clustering.

Kaufman and Rousseeuw (1990) introduced the hierarchical agglomerative clustering to the document domain. Their approach was very classic by comparing pairs of documents for similarity and applying an agglomerative method afterwards. Based on this work, additional research has been done. Zhao and Karypis (2001) investigated similarity measures, Steinbach et al. (2000) analyzed linkage methods and showed that the Unweighted Pair Group with Arithmetic Mean (UWPGAM) is the most accurate one in its category. Benjamin et al. (2003) tackle the problem of large dimensionality in document clustering by using frequent itemssets (Frequent Itemset-based Hierarchical Clustering) which is probably still state of the art.

Flanagan and Himberg (2003) uses the Symbol Clustering Map (SCM) (Flanagan, 2003) for hierarchical clustering of contextual information. It generates higher level context by fusing different sources which provide symbol-like input. The SCM has been developed by the same researcher for this particular task of unsupervised clustering for strings of symbols. It is basically a SOM modified for symbols.

CHAPTER 7

Discussion and outlook

7.1 Discussion

The task of hierarchical clustering is very much connected to define and measure the similarity of the patterns which are to be grouped.

With regard to how to define the similarity of patterns, the lesson from the “ugly duckling” theorem is that it can be only be done with the help of domain knowledge. The similarity has to be based on assumptions about the nature of the data. The assumption used here is that similar objects share common features. For images represented by sensorimotor features the application of this assumption means that similar scenes have a common subset of sensorimotor features. This is an intuitive idea found often in literature.

Under this assumption, Tanimoto and cosine similarity measures are the best suited in theory and in practice. This is due to the fact that they do not distinquish to much (or not at all in the case of cosine measure) between absolute and relative distributions of features. The application to real world object images showed a very slight preference for the Tanimoto similarity, but I would not go as far to call it supperior.

The patterns are represented by sets of sensorimotor features. These triples of feature, action, feature are originally treated like symbols which only allows for comparison for equality. This set representation is transformed into a numerical vector representation with the help of a Self-Organizing Map. The SOM learns the overall set of FAFs by training a set of locally connected neuron which are organized in a map structure. The convergence of the map is supposed to maintain the topology of the input space and map it to output space, even though this can practicly never be achieved without distortions for input data with a higher dimensionality than the output space.

One can probably dispute in how far the training with sensorimotor features can be called biologically inspired. With regard to the mapping of visual as well as motoric information on their own, the connection to similar structures in the brain is given. The original motivation of SOMs lies in the reproduction of the self-organization process found in cortical areas which process sensory inputs. Still the mapping of integrated visual and motoric information into one pattern on the other hand is unclear. One possible interpretation might be to treat it as higher-level structure which learns to recognize sensory input that are temporaly connected.

The quantitative representation for a pattern is then obtained by presenting the FAFs of a scene to the SOM and count which neurons fired. The result is a histogram of the size of the number of neurons which supports the comparison with numerical proximity measures like the before-mentioned ones.

The actual hierarchy building was done with agglomerative hierarchical clustering which a popular approach, used frequently in science. The different linkage rules which determine the order of merging of clusters have an impact on the quality. Of these Ward's linkage rule seems to produce the overall best results.

As a hierarchy assessment measure, the cophenetic correlation is not very useful. The case of already classified objects reveals that there is not much agreement between the CPCC values and the reflection of the class structure in the hierarchy.

The f-measure performs better and can be regarded as a useful measure. The drawback is the requirement of a meaningful class assignment of objects from the start which is usually not given in the task of unsupervised learning in general. Still if there is a flat set of classified labels, it is useful for the assessment of a hierarchy generated for it. This is essentialy the case if for instance images show a certain real-world object from different perspectives, so that there is an intrinsic class for every image.

7.2 Outlook

Future development will be the splitting of a sensorimotor feature into its components in order to separate the visual and motoric information. Those can be trained with individual maps which more resembles the biological model. An integration can afterwards by another layer.

Appendix

APPENDIX A

Implementation

Components

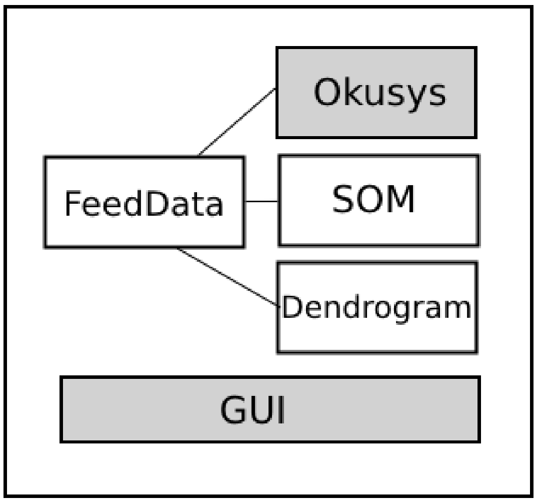


Figure A.1: Components of the application

Figure A.1 shows the components of the system.

- **Okusys** is integrated as an external module. It serves as the data provider. Basically a knowledge base is required for the further steps. Training of Okusys can be triggered as well.
- **SOM** is the Self-Organizing Map module which has been implemented from scratch. It supports rectangluar as well as hexagonal grids. The

neighborhood kernel is fixed to gaussian. Learning rate and neighborhood radius are decaying functions.

- **Dendrogram** generates the hierarchy and has also been implemented from scratch. It is capable of several linkage methods. Furthermore cophenetic correlation and f-measure are computed automatically.
- **FeedData** is a wrappper class for data. It transforms the output from Okusys into a suitable format for the SOM. Furthermore it obtains the data for the dendrogram generation from the SOM.
- **GUI** provides the graphical user interface.

Missing in the figure are utility class for mathematical computation as well as configuration. The `MathUtil` implements for instance similarity measures with all necessary additional functions which are required.

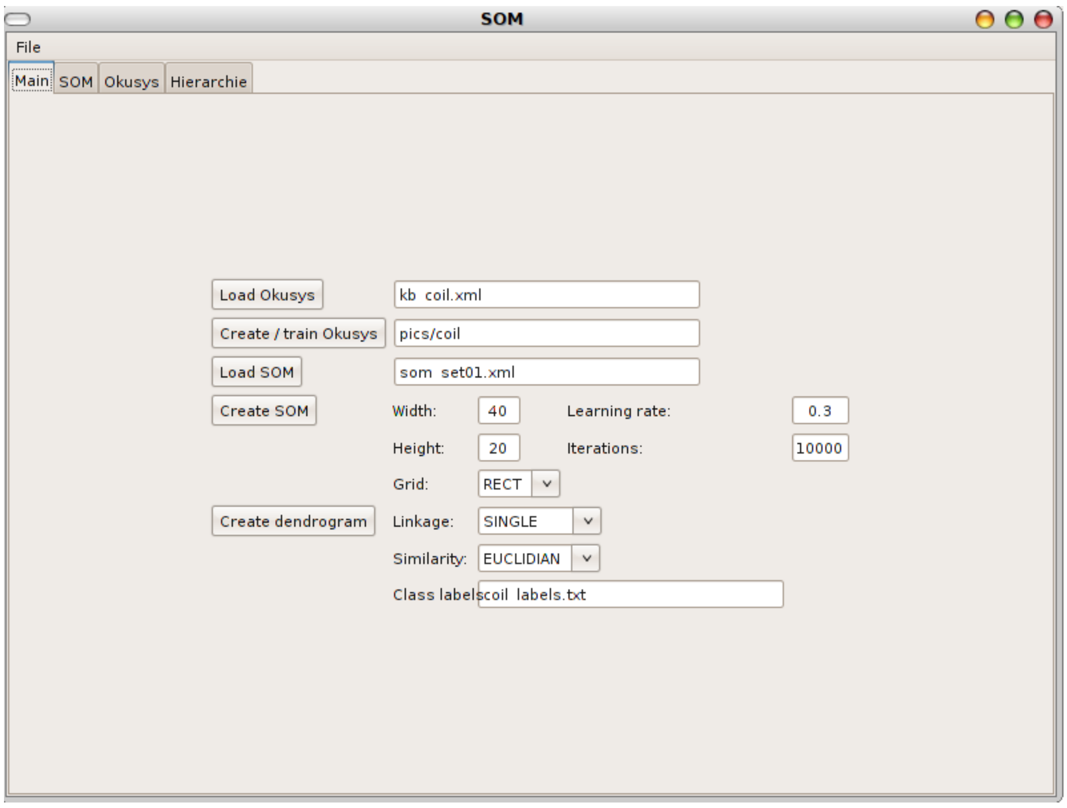
System requirments

The application is written in Java and is thus able to run on any OS which supports Java. The version it has been written and tested with is Java 1.6.

The only factor which poses additional requirments is Okusys. If training of images has to be done, Okusys has to run on Linux (due to IPRS libraries used). Furthermore, the memory of the Java VM has to be set to 256MB or more in order to run.

APPENDIX B

Graphical User Interface



kb coil.xml

pics/coil

som set01.xml

Width: 40

Learning rate: 0.3

Height: 20

Iterations: 10000

Grid: RECT

Linkage: SINGLE

Similarity: EUCLIDIAN

Class labels coil labels.txt

Figure B.1: GUI

The graphical user interface is shown in figure B.1. The necessary components for the clustering can be loaded or created from here. Okusys is required

for the creation of a SOM, a SOM is required for the creation of a dendrogram.
The text fields should point to files (Okusys, SOM, class labels) or to a directory (pictures). Note that if Okusys or SOM are created, the paths are used for saving.

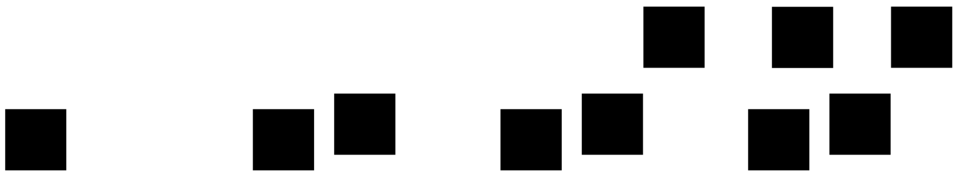
SOM options Width, heigh, learning rate, grid type and the number of iterations can be entered here. Valid values are required. Note that the iterations figure means number of *iterations of the input set*, and not the total number.

Dendrogram options The dendrogram options are basically the linkage rule as well as the similarity measure. The class labels are not required, but can be stated for the calculation of the f-measure.

APPENDIX C

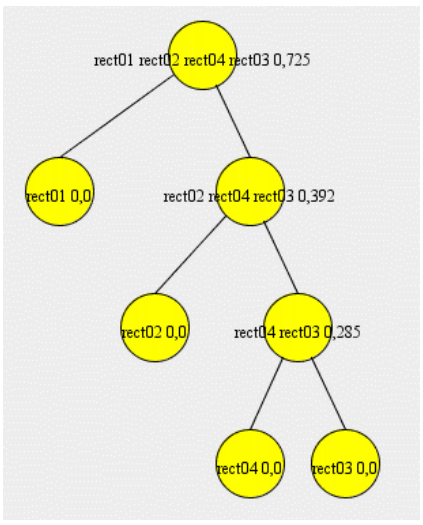
Test outputs

C.1 Set 01

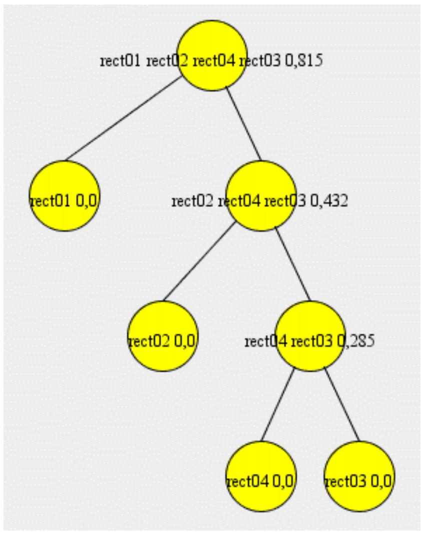


(a) rect01 (b) rect02 (c) rect03 (d) rect04

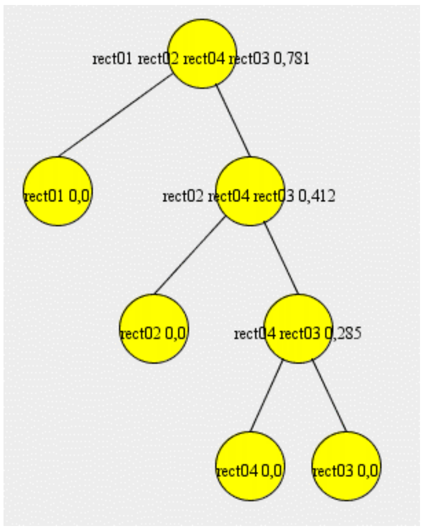
Figure C.1: Test set 01



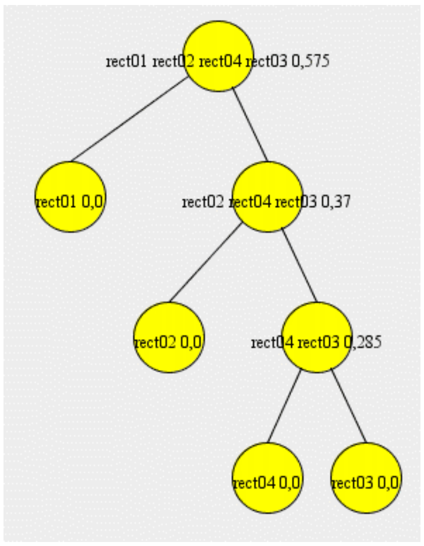
(a) Single



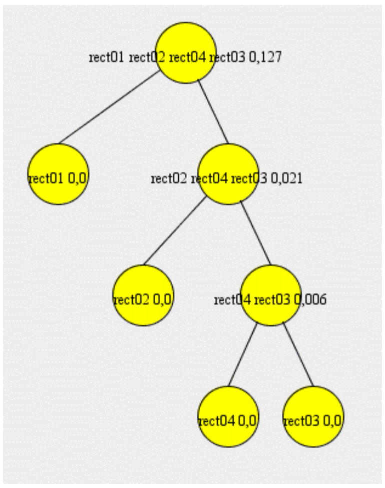
(b) Complete



(c) Average pair group

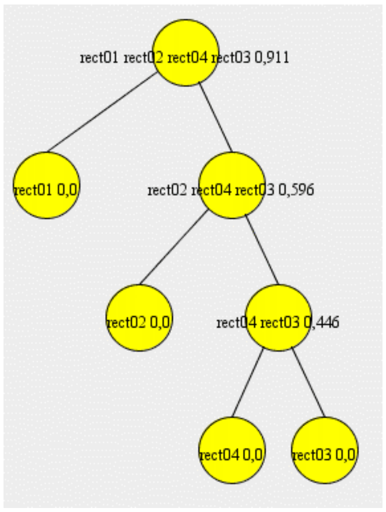


(d) Average within group

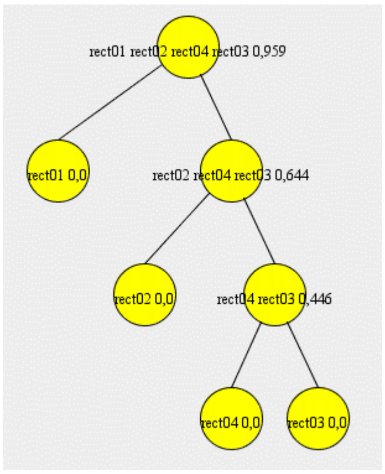


(e) Ward

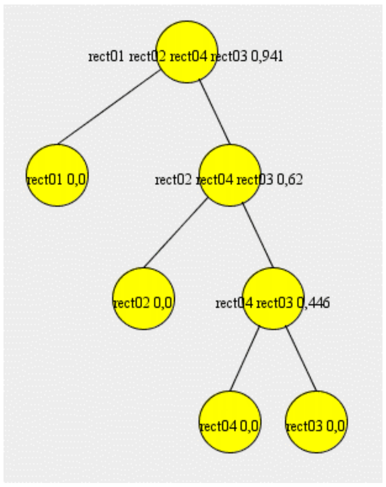
Figure C.2: Output of set01 with cosine and different linkages



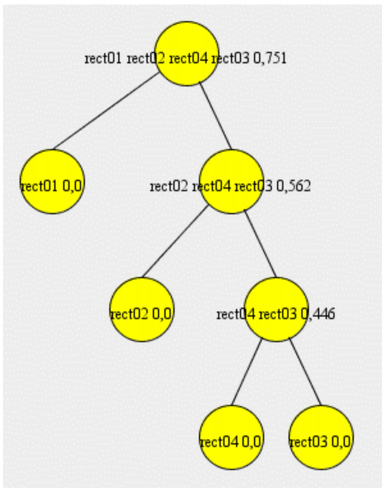
(a) Single



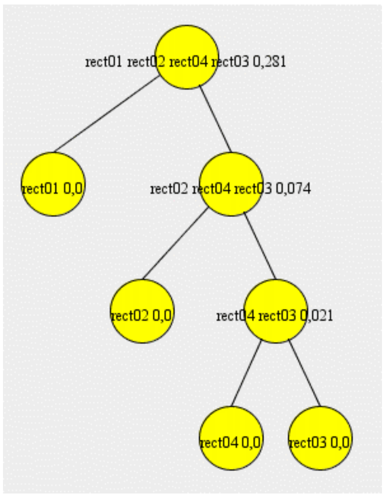
(b) Complete



(c) Average pair group

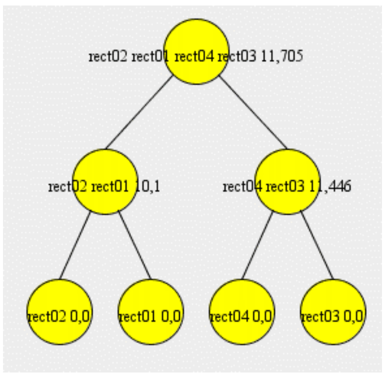


(d) Average within group

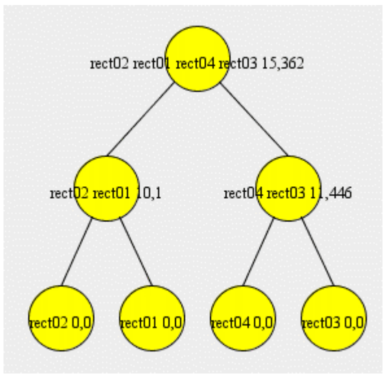


(e) Ward

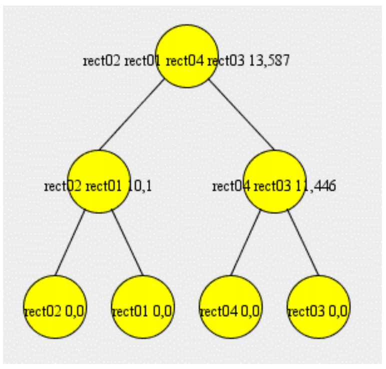
Figure C.3: Output of set01 with Tanimoto and different linkages



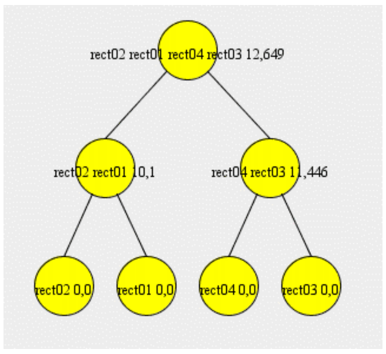
(a) Single



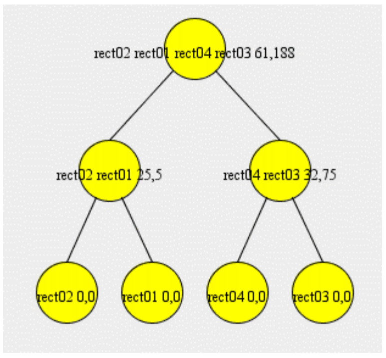
(b) Complete



(c) Average pair group



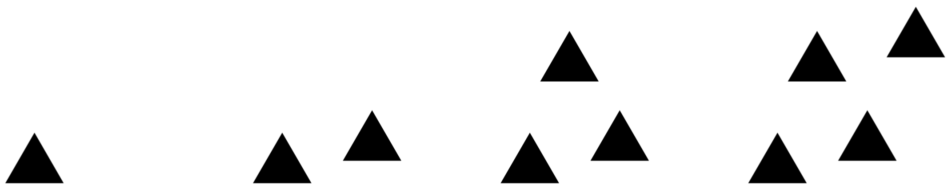
(d) Average within group



(e) Ward

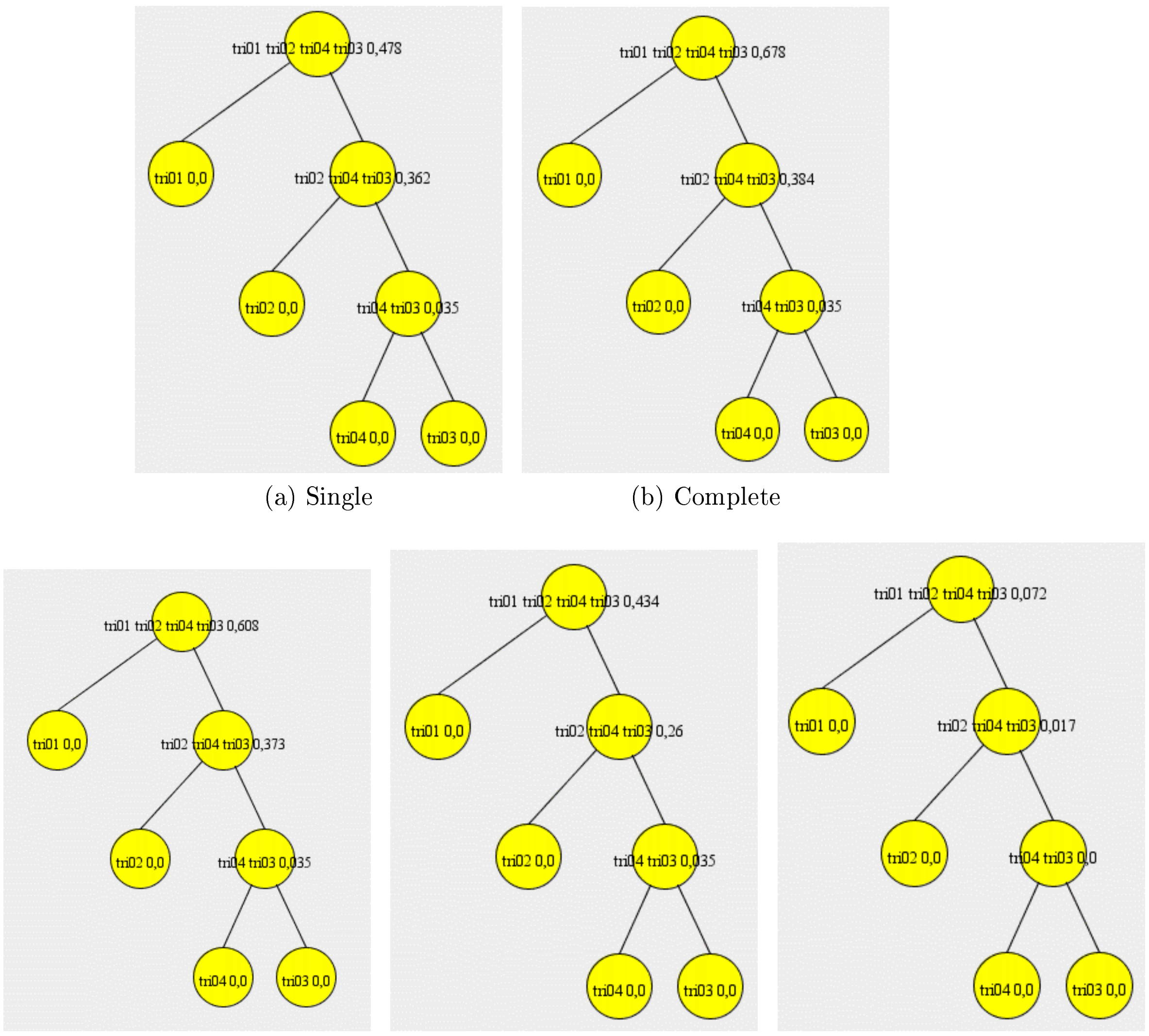
Figure C.4: Output of set01 with Euclidian and different linkages

C.2 Set 02



(a) tri01 (b) tri02 (c) tri03 (d) tri04

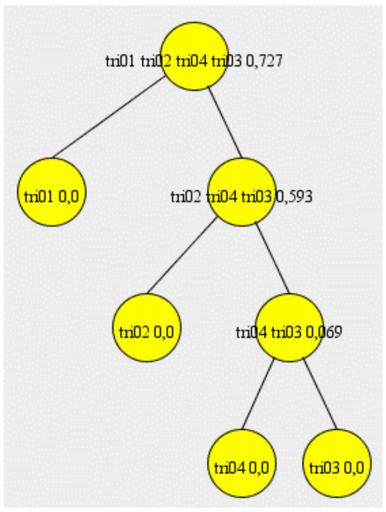
Figure C.5: Test set 02



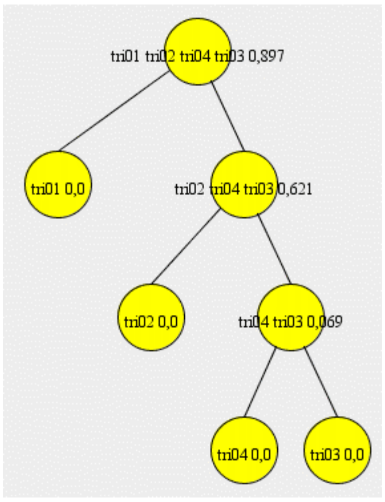
(a) Single (b) Complete

(c) Average pair group (d) Average within group (e) Ward

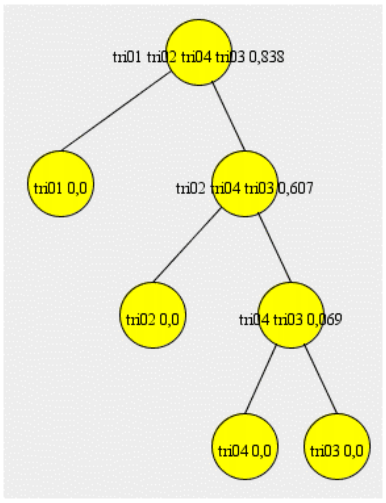
Figure C.6: Output of set02 with cosine and different linkages



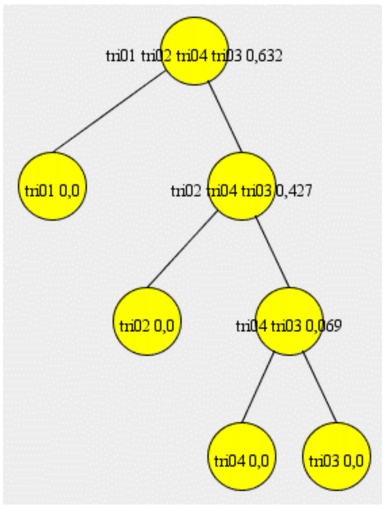
(a) Single



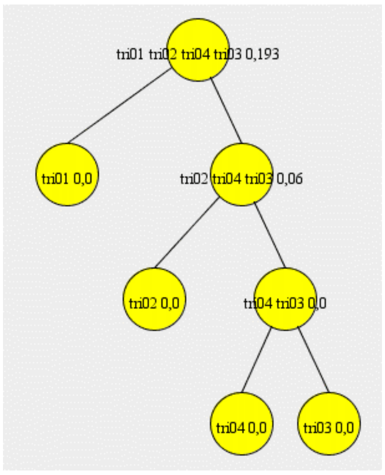
(b) Complete



(c) Average pair group

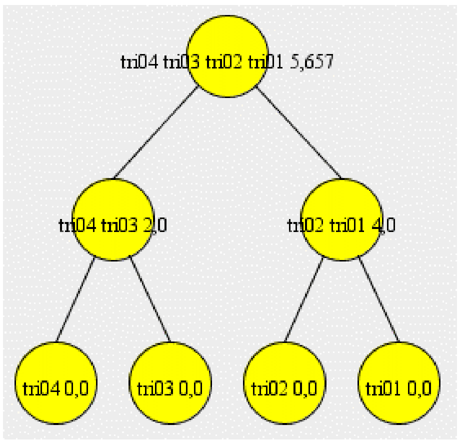


(d) Average within group

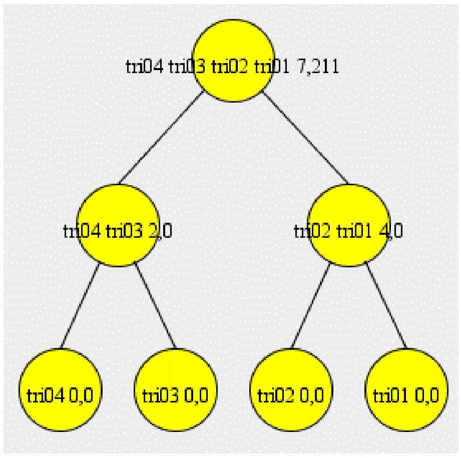


(e) Ward

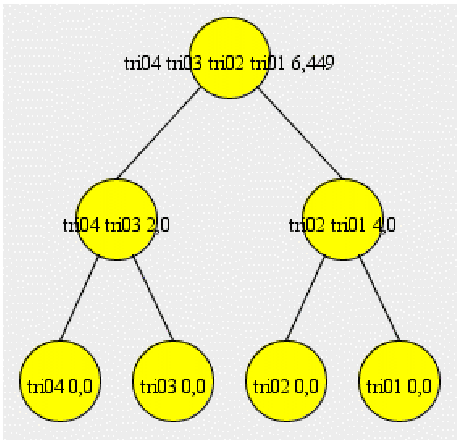
Figure C.7: Output of set02 with Tanimoto and different linkages



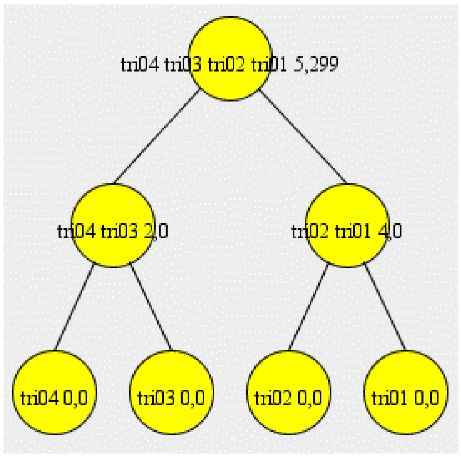
(a) Single



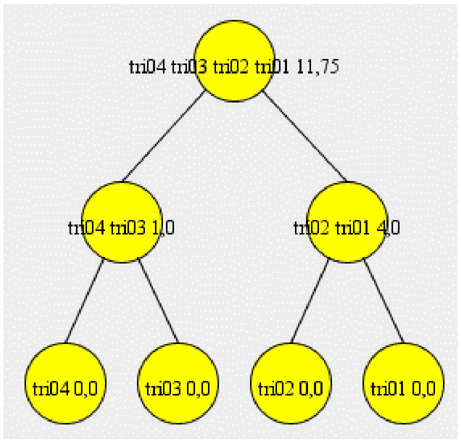
(b) Complete



(c) Average pair group



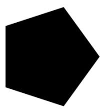
(d) Average within group

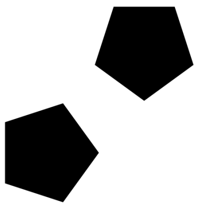


(e) Ward

Figure C.8: Output of set02 with Euclidian and different linkages

C.3 Set 03


(a) penta01


(b) penta02

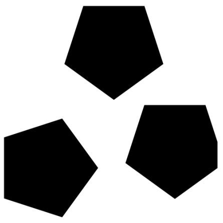
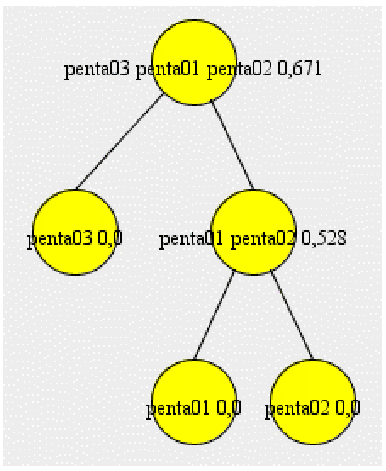
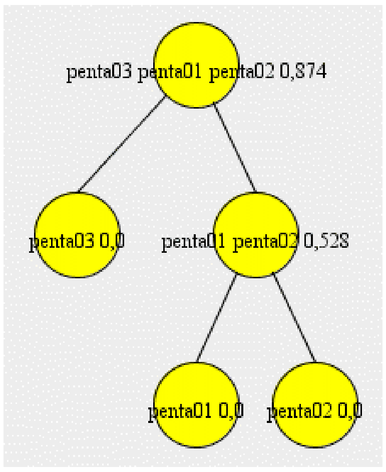

(c) penta03

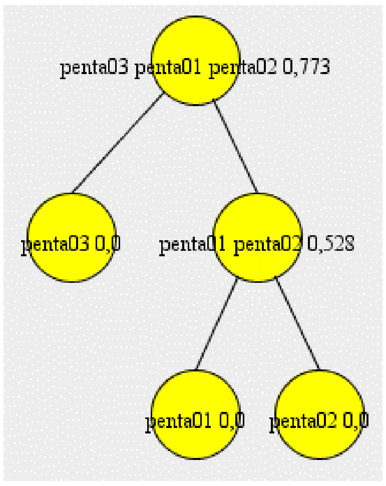
Figure C.9: Test set 03



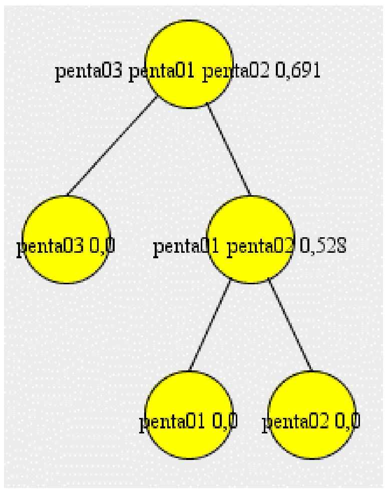
(a) Single



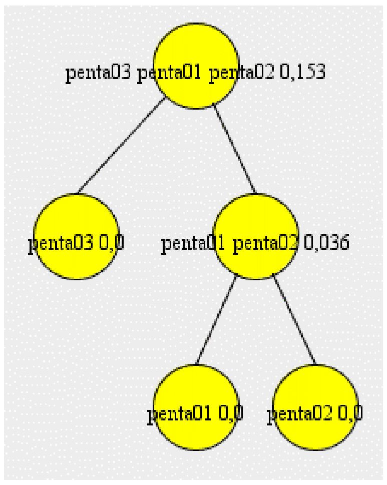
(b) Complete



(c) Average pair group

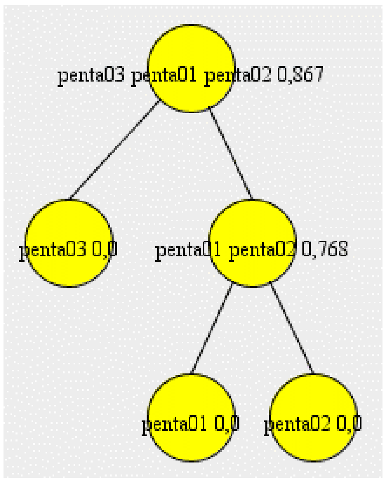


(d) Average within group

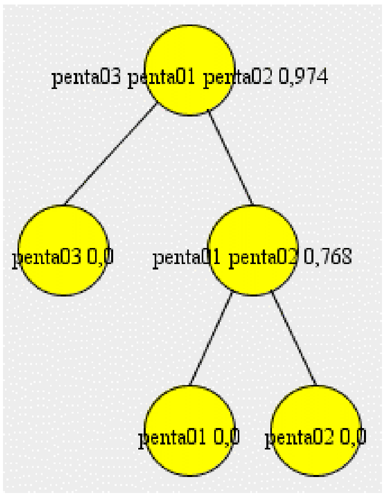


(e) Ward

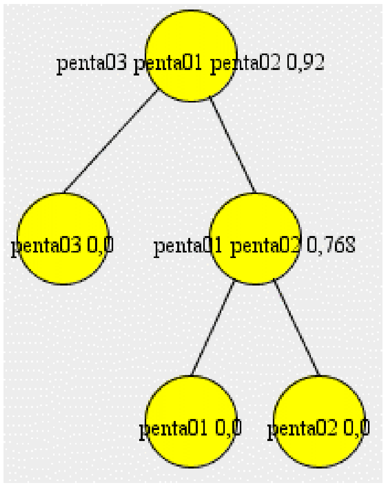
Figure C.10: Output of set03 with cosine and different linkages (cont.)



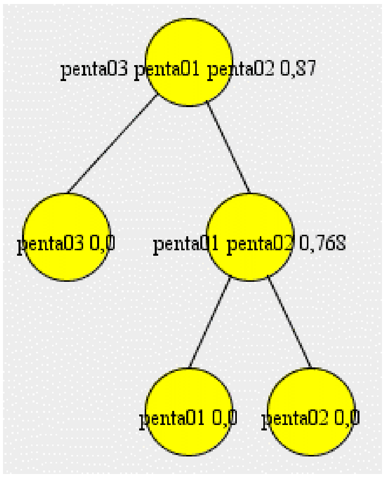
(a) Single



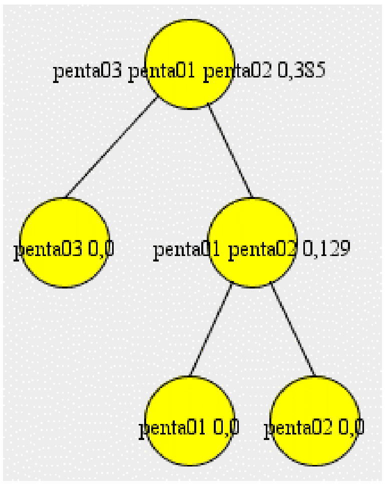
(b) Complete



(c) Average pair group

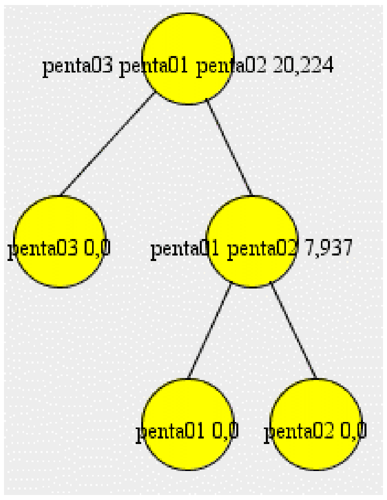


(d) Average within group

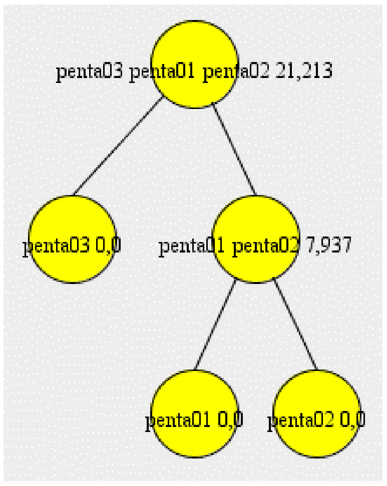


(e) Ward

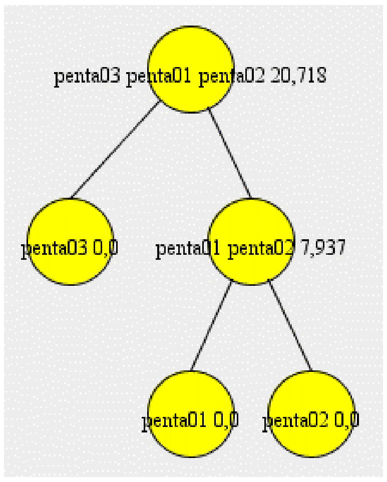
Figure C.11: Output of set03 with Tanimoto and different linkages



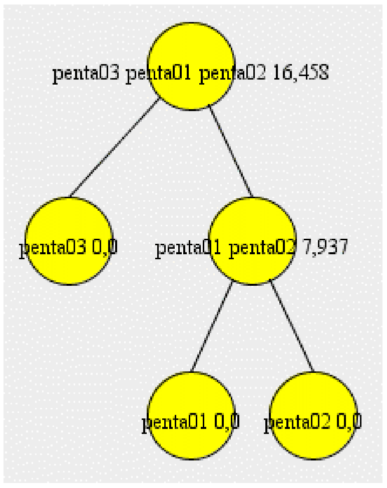
(a) Single



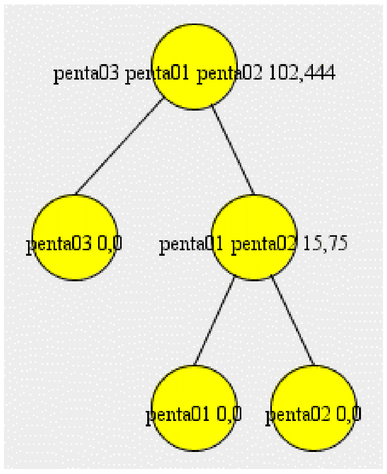
(b) Complete



(c) Average pair group



(d) Average within group



(e) Ward

Figure C.12: Output of set03 with Euclidian and different linkages

C.4 Set 04

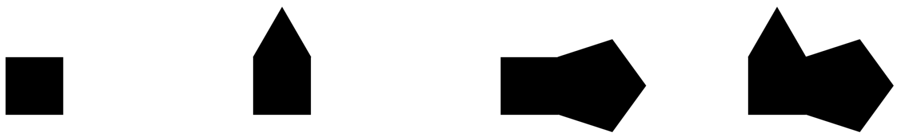
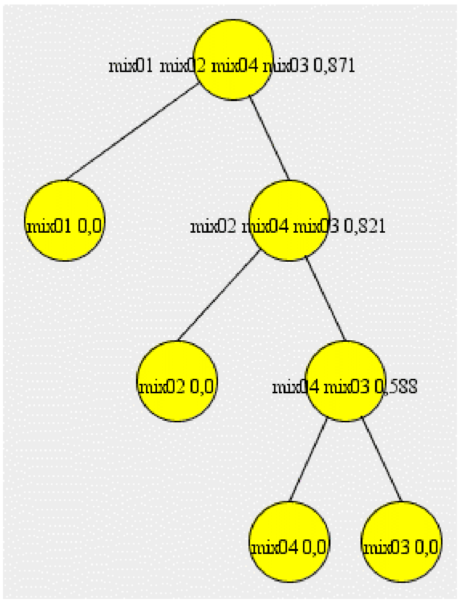
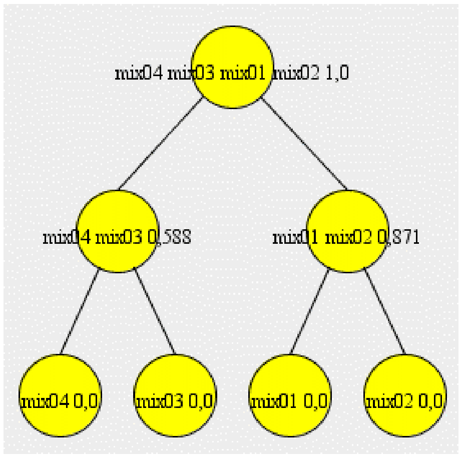


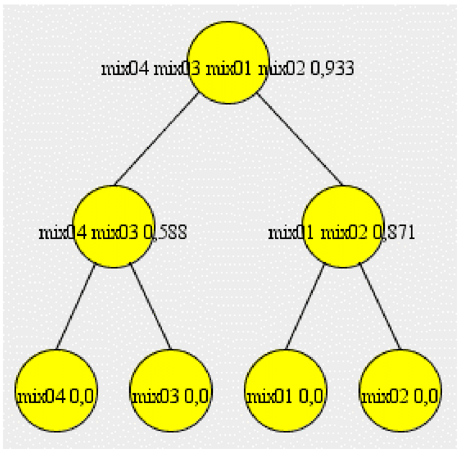
Figure C.13: Test set 04



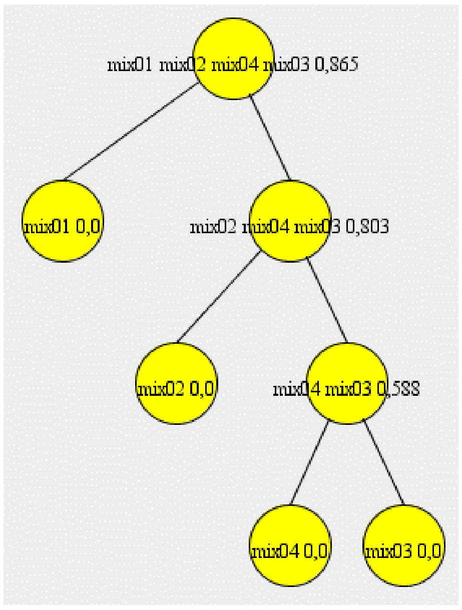
(a) Single



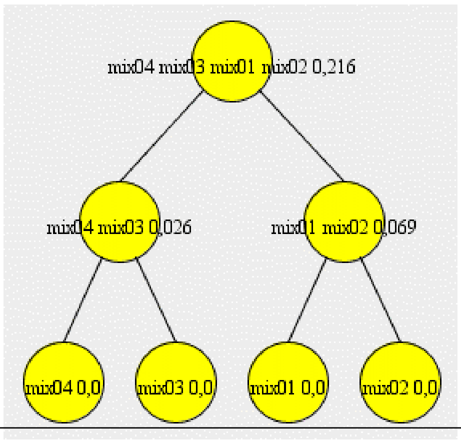
(b) Complete



(c) Average pair group

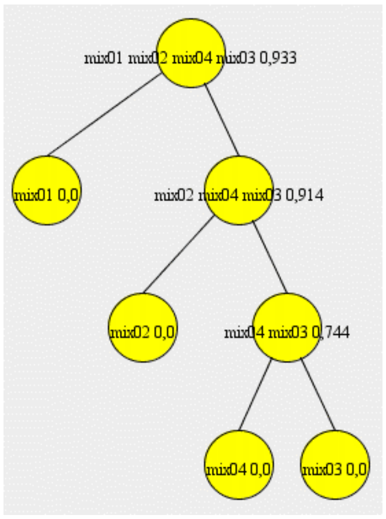


(d) Average within group

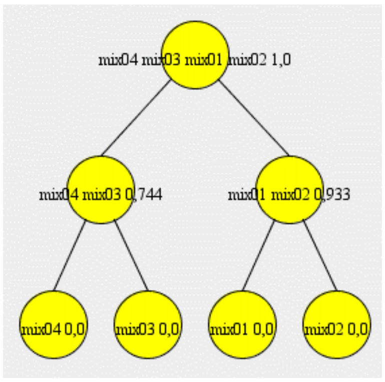


(e) Ward

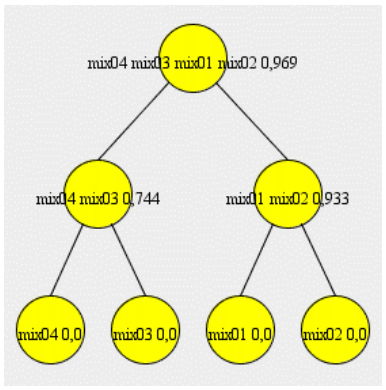
Figure C.14: Output of set04 with cosine and different linkages



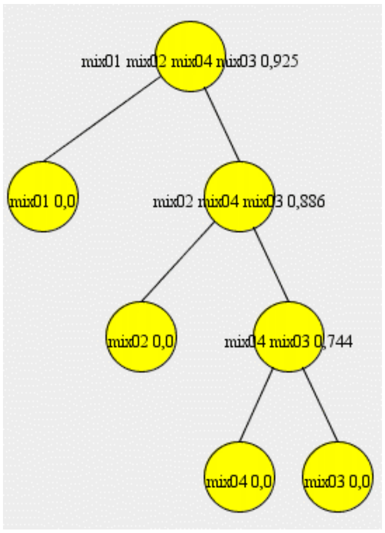
(a) Single



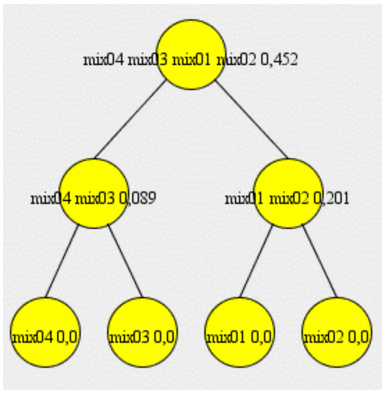
(b) Complete



(c) Average pair group

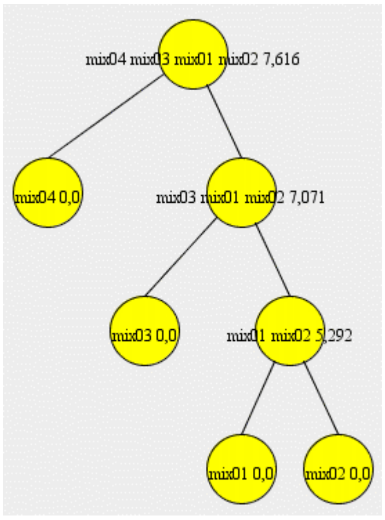


(d) Average within group

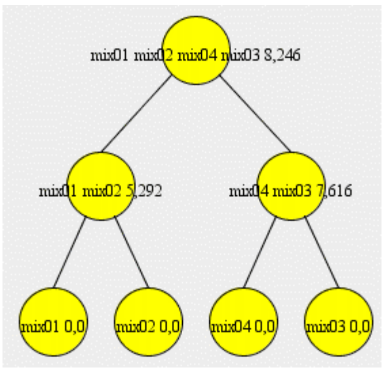


(e) Ward

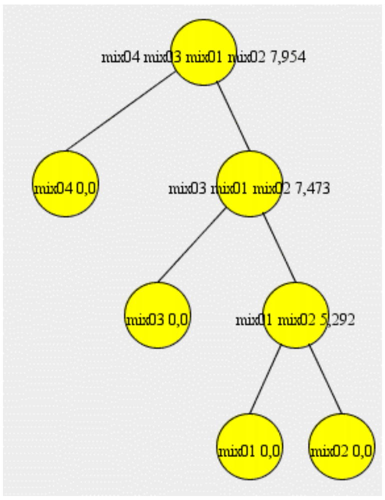
Figure C.15: Output of set04 with Tanimoto and different linkages



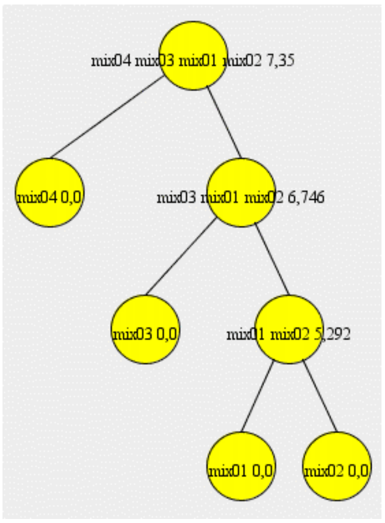
(a) Single



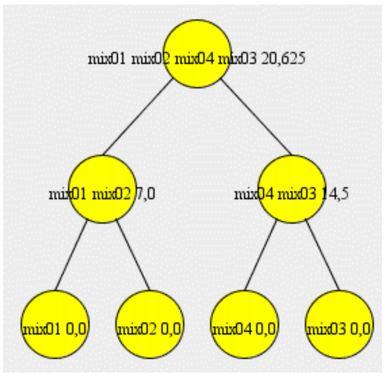
(b) Complete



(c) Average pair group



(d) Average within group

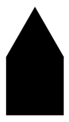


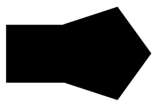
(e) Ward

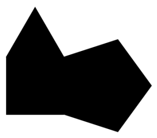
Figure C.16: Output of set04 with Euclidian and different linkages

C.5 Set 05

(a) mix01


(b) mix02


(c) mix03


(d) mix04

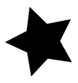
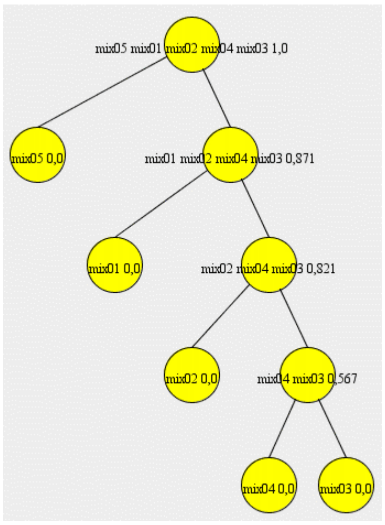
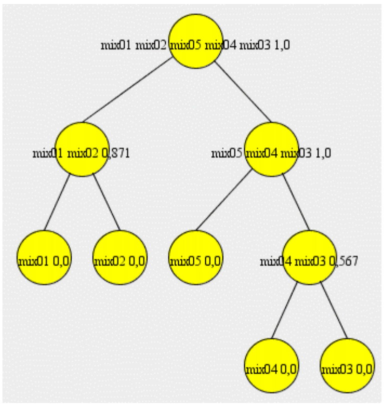

(e) mix05

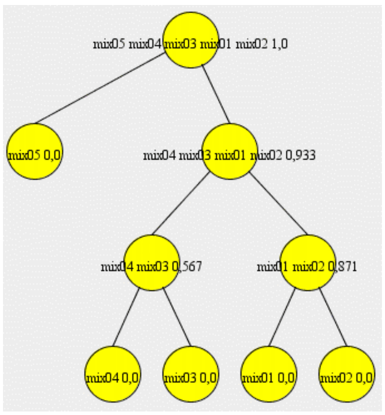
Figure C.17: Test set 05



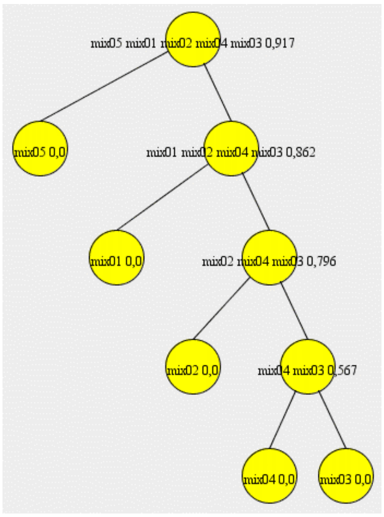
(a) Single



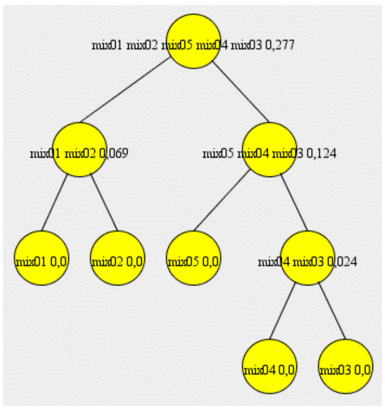
(b) Complete



(c) Average pair group

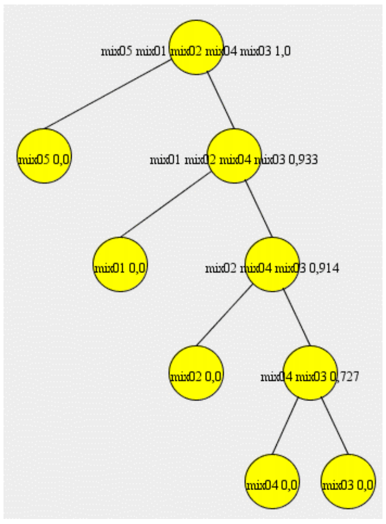


(d) Average within group

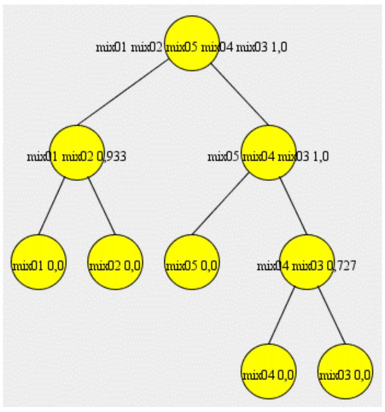


(e) Ward

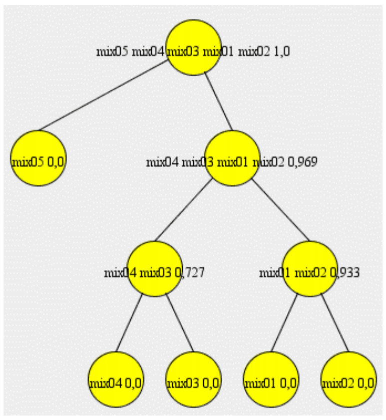
Figure C.18: Output of set05 with cosine and different linkages



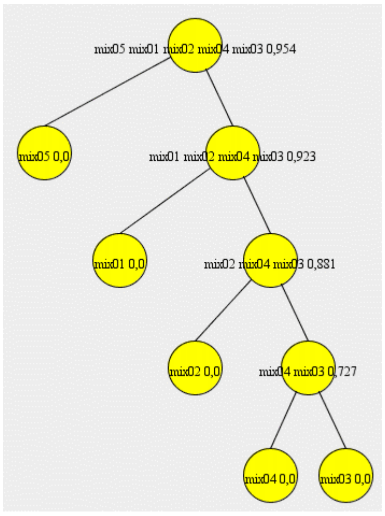
(a) Single



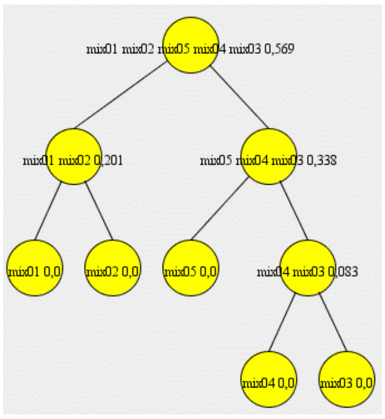
(b) Complete



(c) Average pair group

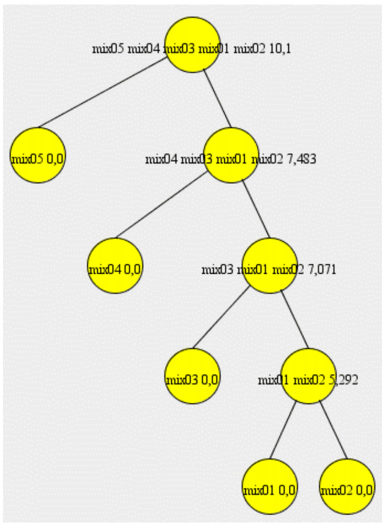


(d) Average within group

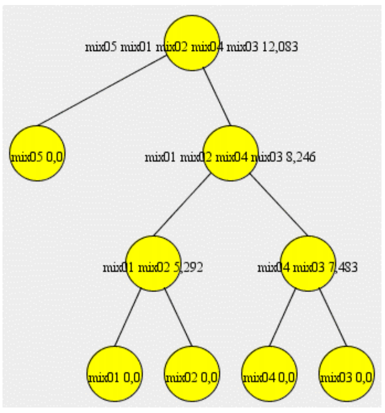


(e) Ward

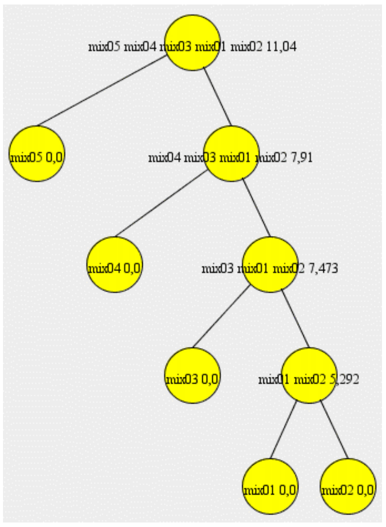
Figure C.19: Output of set05 with Tanimoto and different linkages



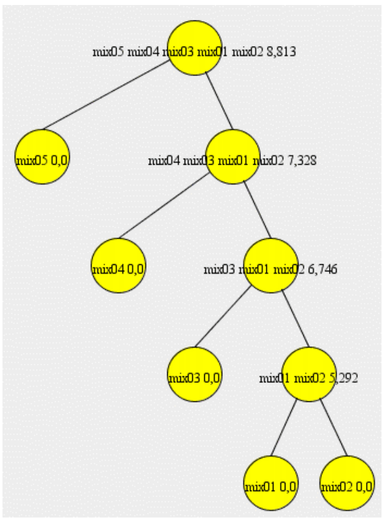
(a) Single



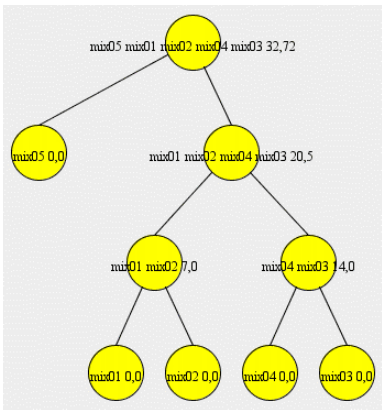
(b) Complete



(c) Average pair group



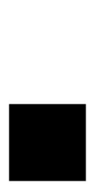
(d) Average within group

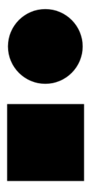


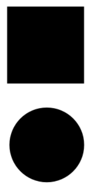
(e) Ward

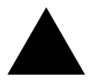
Figure C.20: Output of set05 with Euclidian and different linkages

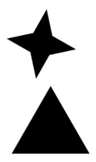
C.6 Set 06


(a) rect01


(b) rect02


(c) rect03


(d) tri01


(e) tri02

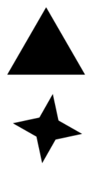
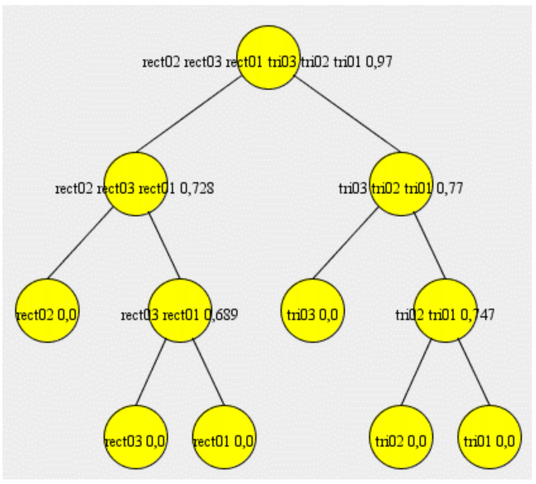
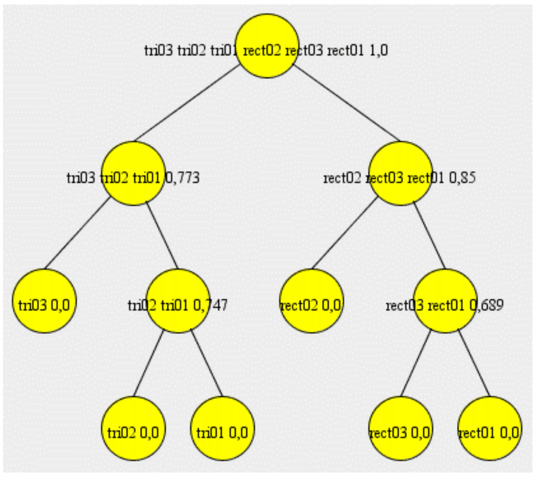

(f) tri03

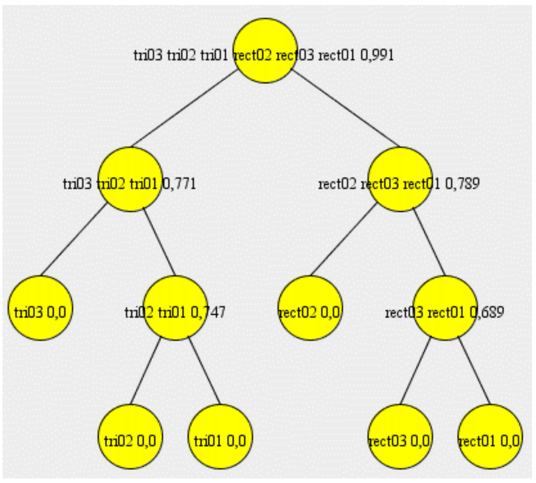
Figure C.21: Test set 06



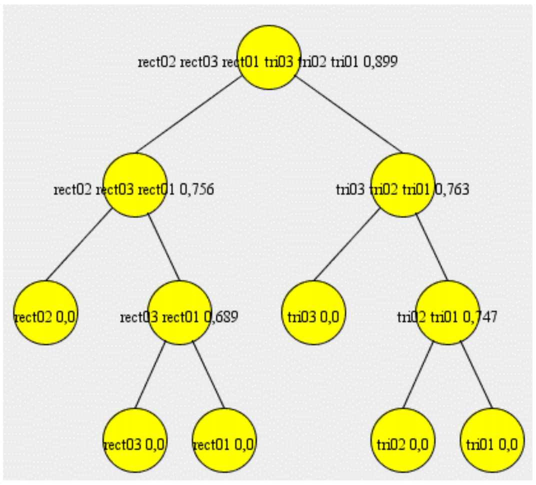
(a) Single



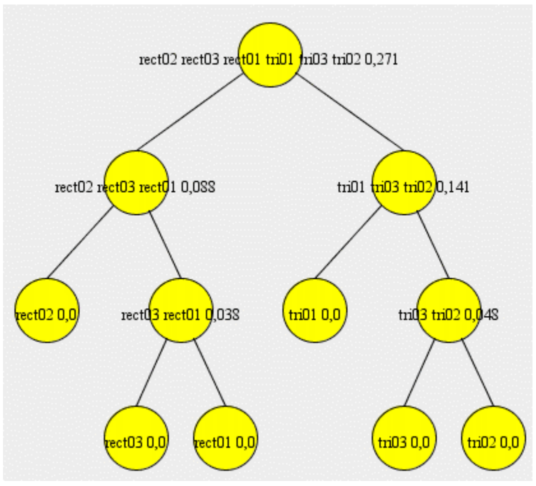
(b) Complete



(c) Average pair group

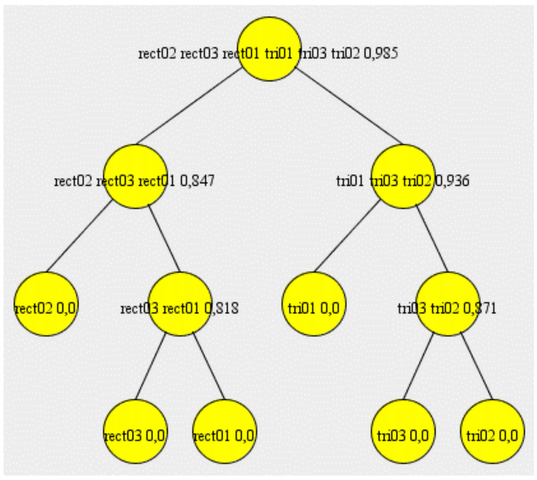


(d) Average within group

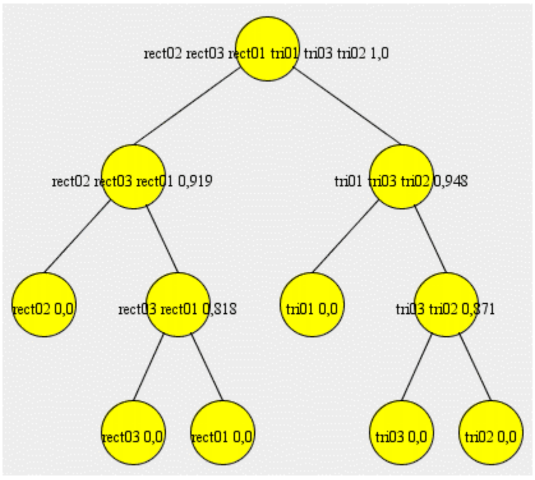


(e) Ward

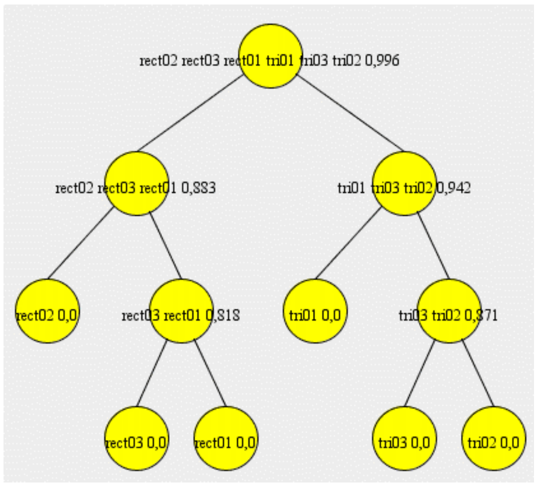
Figure C.22: Output of set06 with cosine and different linkages



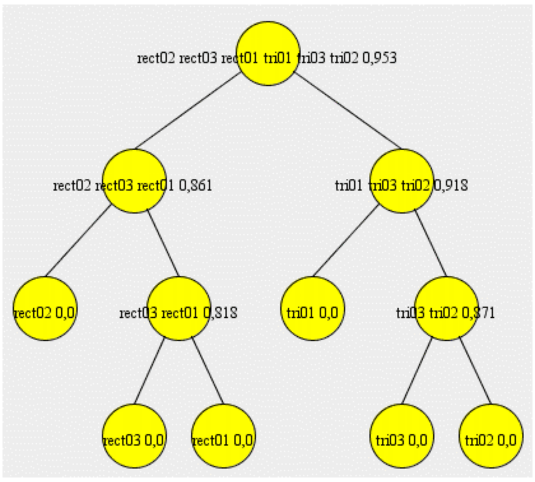
(a) Single



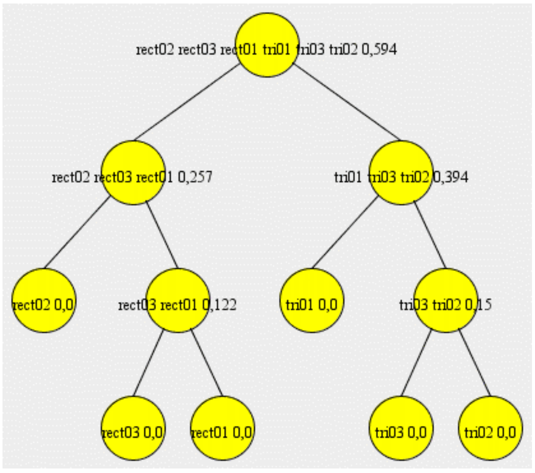
(b) Complete



(c) Average pair group

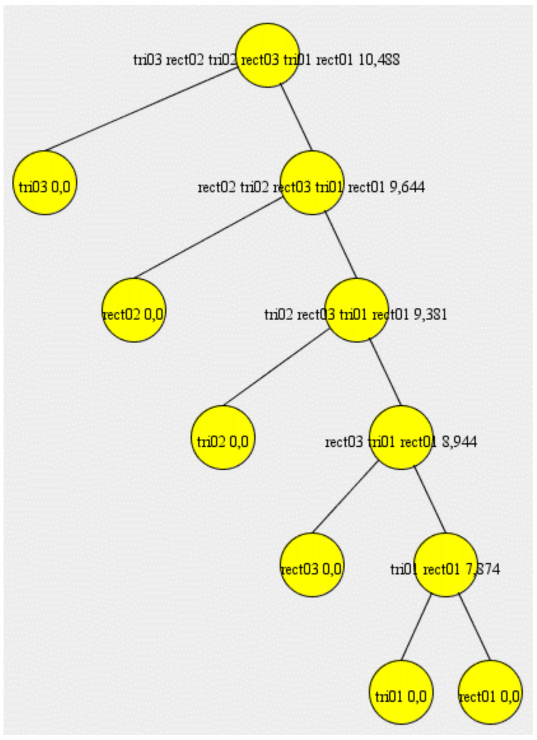


(d) Average within group

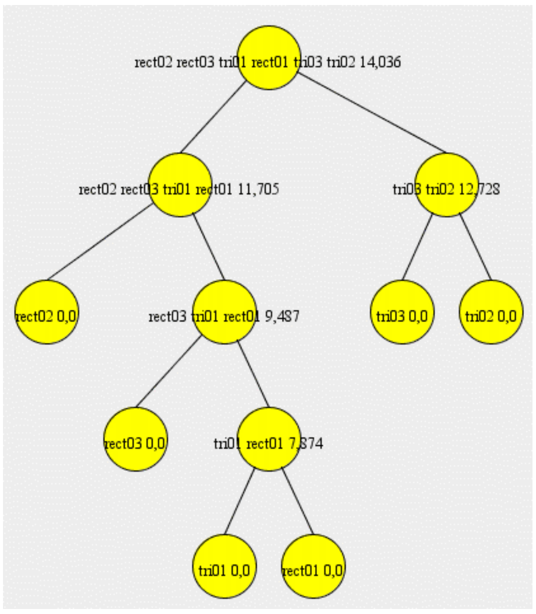


(e) Ward

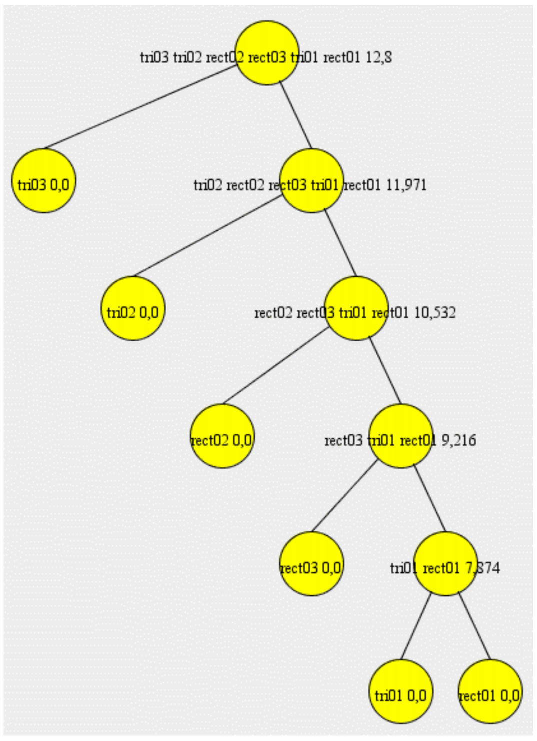
Figure C.23: Output of set06 with Tanimoto and different linkages


tri03 rect02 tri02 rect03 tri01 rect01 10,488
tri03 0,0
rect02 tri02 rect03 tri01 rect01 9,644
rect02 0,0
tri02 rect03 tri01 rect01 9,381
tri02 0,0
rect03 tri01 rect01 8,944
rect03 0,0
tri01 rect01 7,874
tri01 0,0
rect01 0,0

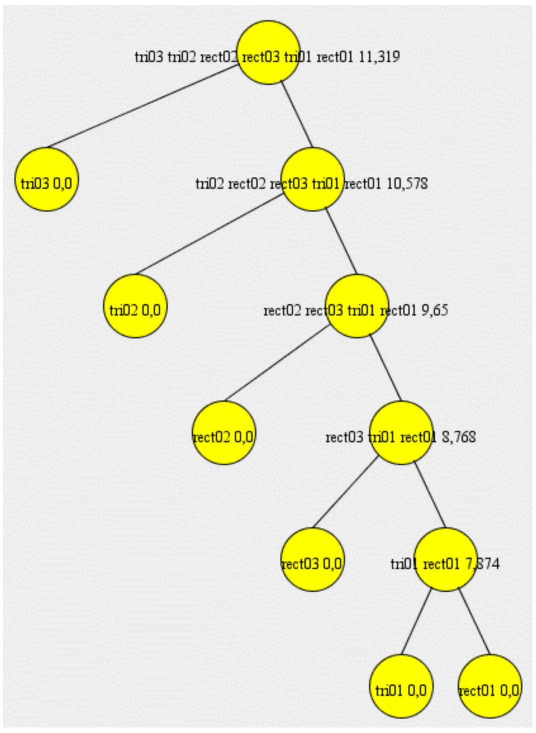
(a) Single


rect02 rect03 tri01 rect01 tri03 tri02 14,036
rect02 rect03 tri01 rect01 11,705
tri03 tri02 12,728
rect02 0,0
rect03 tri01 rect01 9,487
tri03 0,0
tri02 0,0
rect03 0,0
tri01 rect01 7,874
tri01 0,0
rect01 0,0

(b) Complete

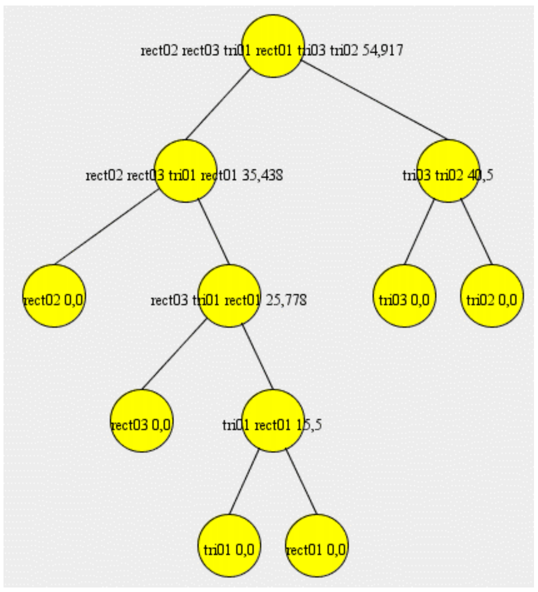

tri03 tri02 rect02 rect03 tri01 rect01 12,8
tri03 0,0
tri02 rect02 rect03 tri01 rect01 11,971
tri02 0,0
rect02 rect03 tri01 rect01 10,532
rect02 0,0
rect03 tri01 rect01 9,216
rect03 0,0
tri01 rect01 7,874
tri01 0,0
rect01 0,0

(c) Average pair group


tri03 tri02 rect02 rect03 tri01 rect01 11,319
tri03 0,0
tri02 rect02 rect03 tri01 rect01 10,578
tri02 0,0
rect02 rect03 tri01 rect01 9,65
rect02 0,0
rect03 tri01 rect01 8,768
rect03 0,0
tri01 rect01 7,874
tri01 0,0
rect01 0,0

(d) Average within group

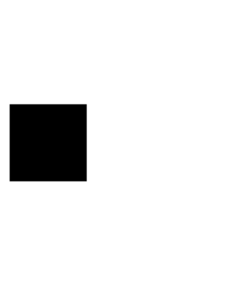
Figure C.24: Output of set06 with Euclidian and different linkages

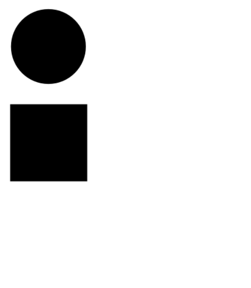

rect02 rect03 tri01 rect01 tri03 tri02 54,917
rect02 rect03 tri01 rect01 35,438
tri03 tri02 40,5
rect02 0,0
rect03 tri01 rect01 25,778
tri03 0,0
tri02 0,0
rect03 0,0
tri01 rect01 15,5
tri01 0,0
rect01 0,0

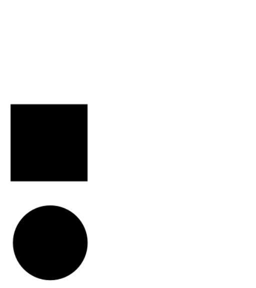
(e) Ward

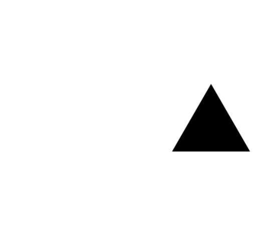
Figure C.24: Output of set06 with Euclidian and different linkages (cont.)

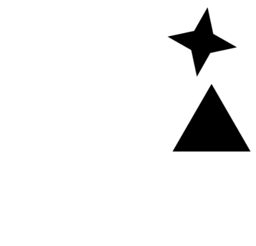
C.7 Set 07

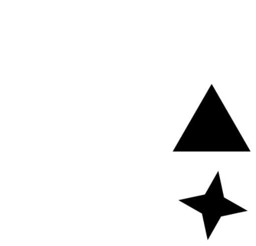

(a) rect01

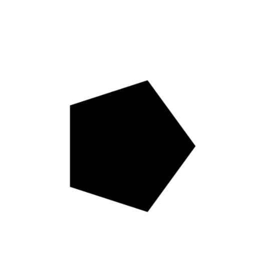

(b) rect02

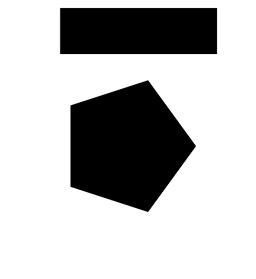

(c) rect03


(d) tri01


(e) tri02


(f) tri03


(g) pent01


(h) pent02

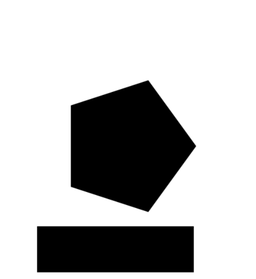
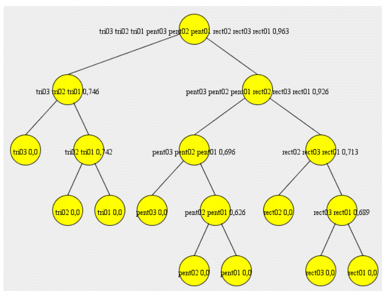
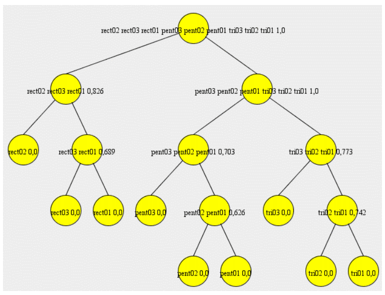

(i) pent03

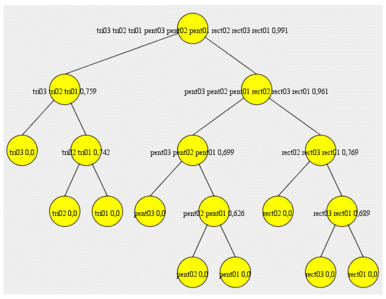
Figure C.25: Test set 07



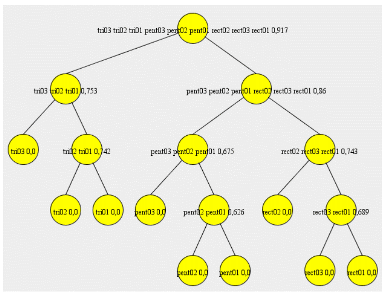
(a) Single



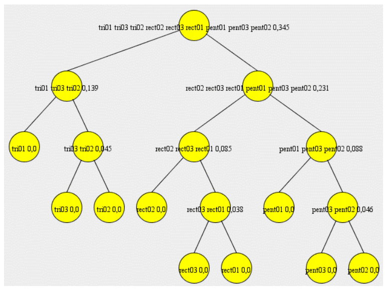
(b) Complete



(c) Average pair group

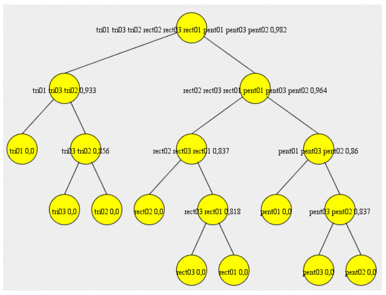


(d) Average within group

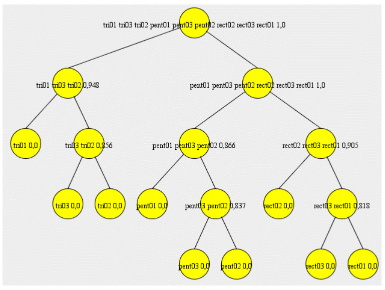


(e) Ward

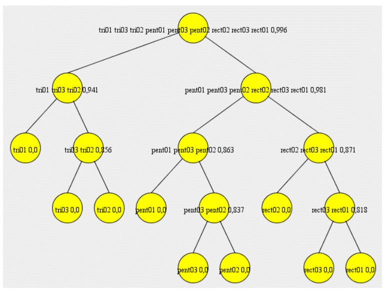
Figure C.26: Output of set07 with cosine and different linkages



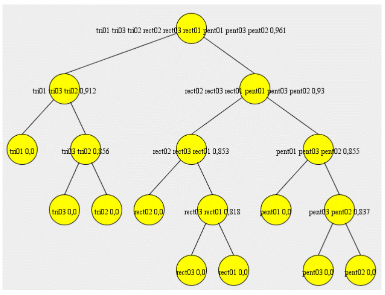
(a) Single



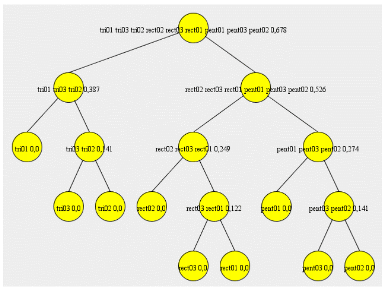
(b) Complete



(c) Average pair group

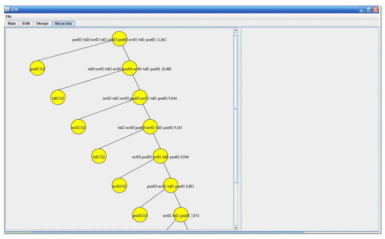


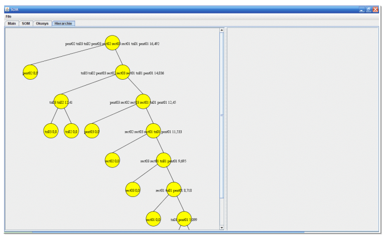
(d) Average within group

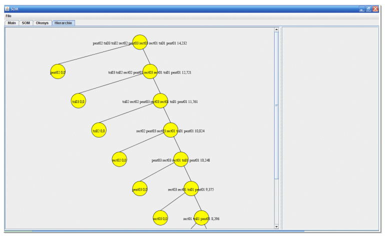


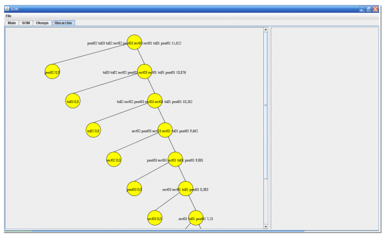
(e) Ward

Figure C.27: Output of set07 with Tanimoto and different linkages


(a) Single


(b) Complete


(c) Average pair group


(d) Average within group

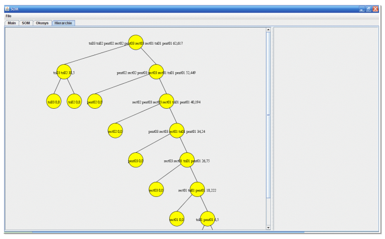

(e) Ward

Figure C.28: Output of set07 with Euclidian and different linkages

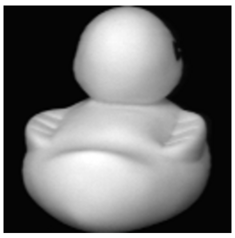
C.8 Set Coil

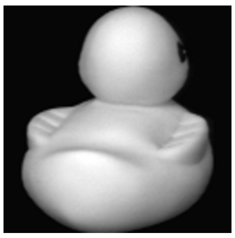
(a) obj16 2

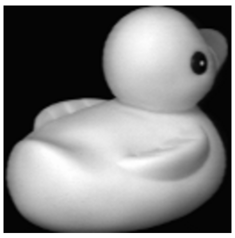
(b) obj16 20

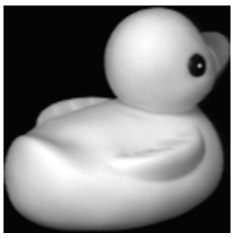
(c) obj16 21

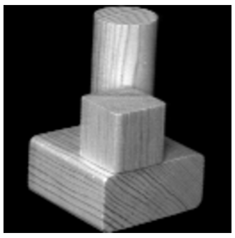
(d) obj16 22

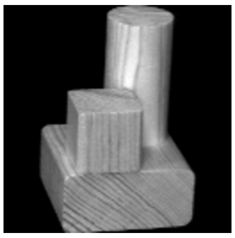

(e) obj1 20

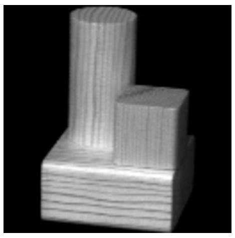

(f) obj1 21


(g) obj1 27


(h) obj1 28


(i) obj7 10


(j) obj7 15


(k) obj7 2

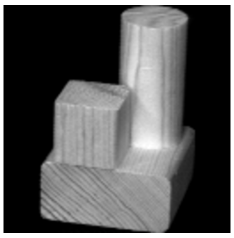
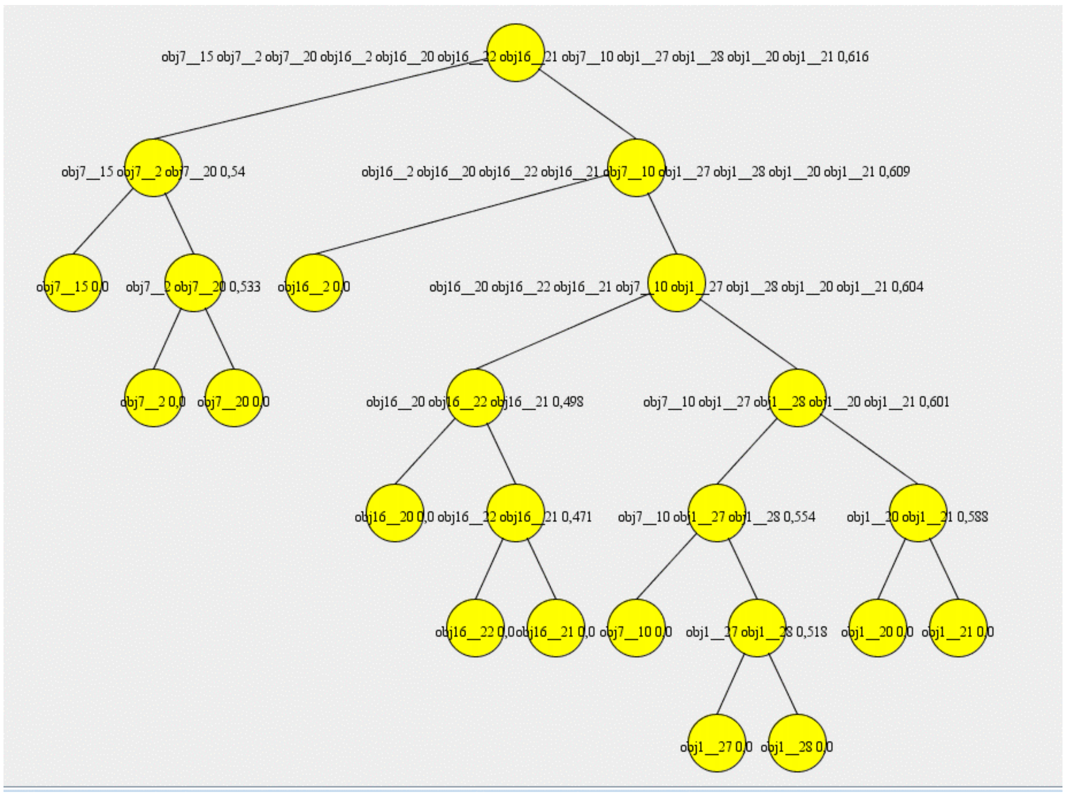
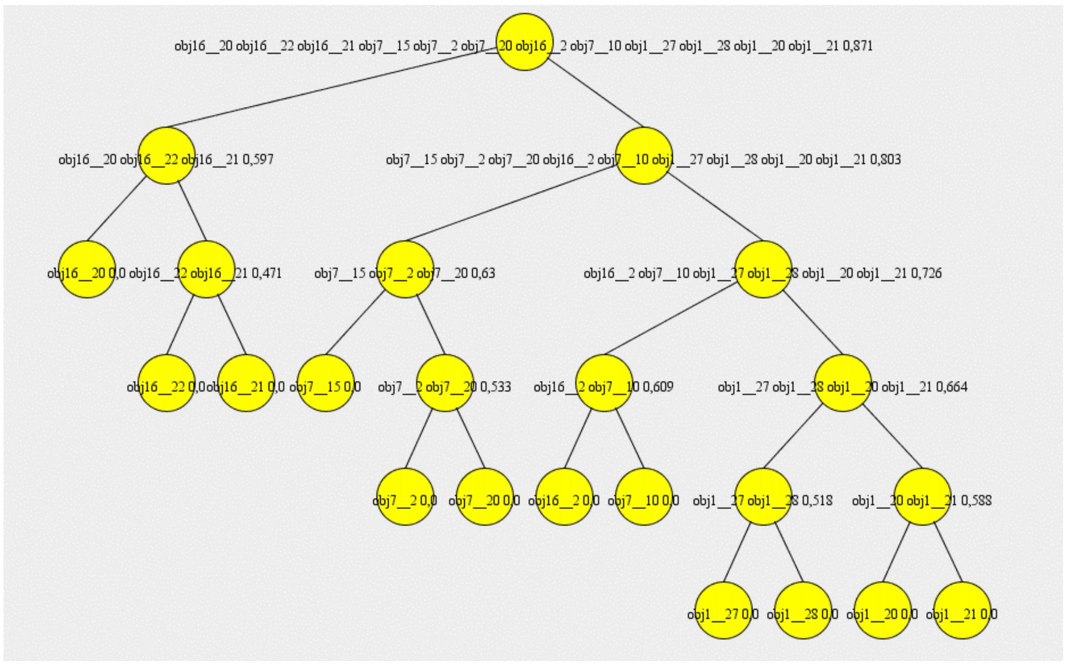

(l) obj7 20

Figure C.29: Test set Coil

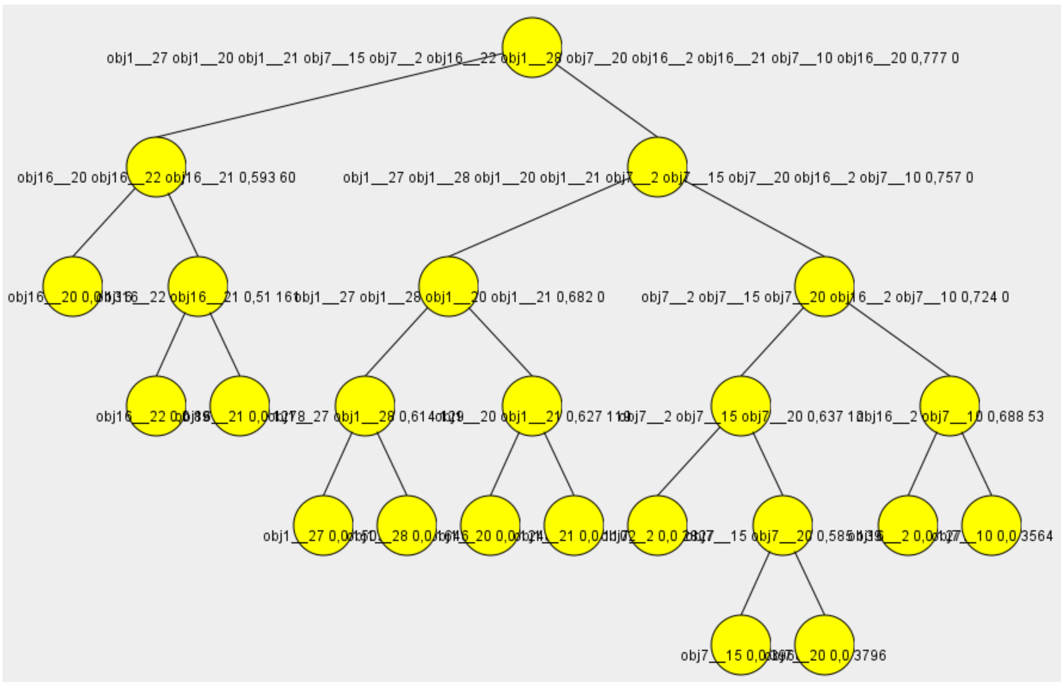

obj7__15 obj7__2 obj7__20 obj16__2 obj16__20 obj16__22 obj16__21 obj7__10 obj1__27 obj1__28 obj1__20 obj1__21 0,616
obj7__15 obj7__2 obj7__20 0,54
obj16__2 obj16__20 obj16__22 obj16__21 obj7__10 obj1__27 obj1__28 obj1__20 obj1__21 0,609
obj7__15 0,0
obj7__2 obj7__20 0,533
obj16__2 0,0
obj16__20 obj16__22 obj16__21 obj7__10 obj1__27 obj1__28 obj1__20 obj1__21 0,604
obj7__2 0,0
obj7__20 0,0
obj16__20 obj16__22 obj16__21 0,498
obj7__10 obj1__27 obj1__28 obj1__20 obj1__21 0,601
obj16__20 0,0
obj16__22 obj16__21 0,471
obj7__10 obj1__27 obj1__28 0,554
obj1__20 obj1__21 0,588
obj16__22 0,0
obj16__21 0,0
obj7__10 0,0
obj1__27 obj1__28 0,518
obj1__20 0,0
obj1__21 0,0
obj1__27 0,0
obj1__28 0,0

(a) Single

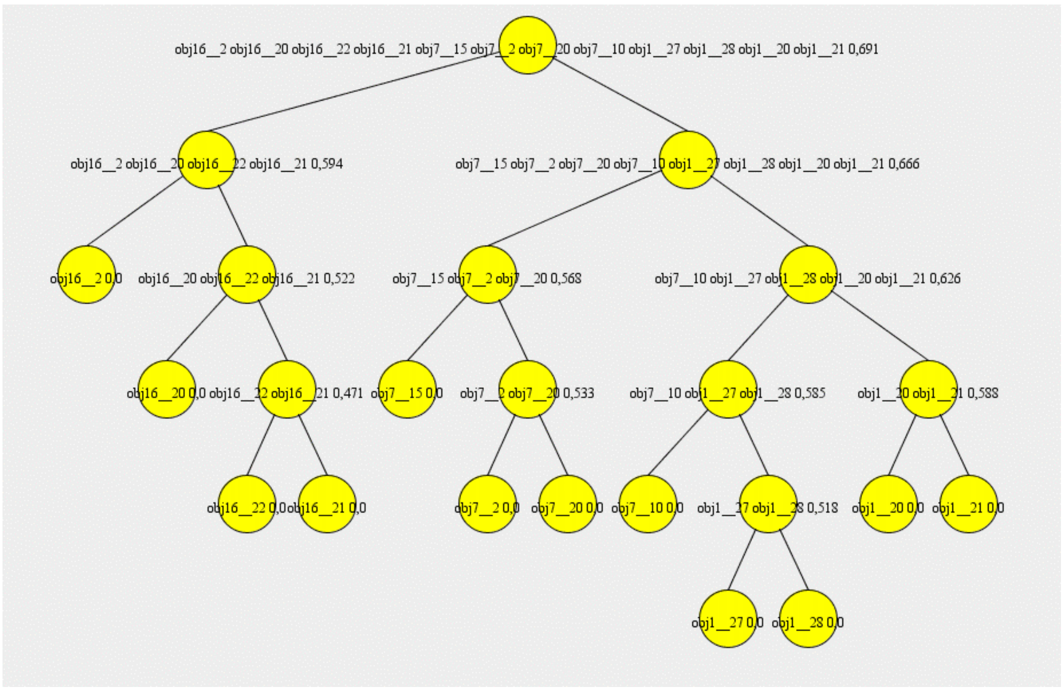

obj16__20 obj16__22 obj16__21 obj7__15 obj7__2 obj7__20 obj16__2 obj7__10 obj1__27 obj1__28 obj1__20 obj1__21 0,871
obj16__20 obj16__22 obj16__21 0,597
obj7__15 obj7__2 obj7__20 obj16__2 obj7__10 obj1__27 obj1__28 obj1__20 obj1__21 0,803
obj16__20 0,0
obj16__22 obj16__21 0,471
obj7__15 obj7__2 obj7__20 0,63
obj16__2 obj7__10 obj1__27 obj1__28 obj1__20 obj1__21 0,726
obj16__22 0,0
obj16__21 0,0
obj7__15 0,0
obj7__2 obj7__20 0,533
obj16__2 obj7__10 0,609
obj1__27 obj1__28 obj1__20 obj1__21 0,664
obj7__2 0,0
obj7__20 0,0
obj16__2 0,0
obj7__10 0,0
obj1__27 obj1__28 0,518
obj1__20 obj1__21 0,588
obj1__27 0,0
obj1__28 0,0
obj1__20 0,0
obj1__21 0,0

(b) Complete

Figure C.30: Output of coil set with cosine and different linkages

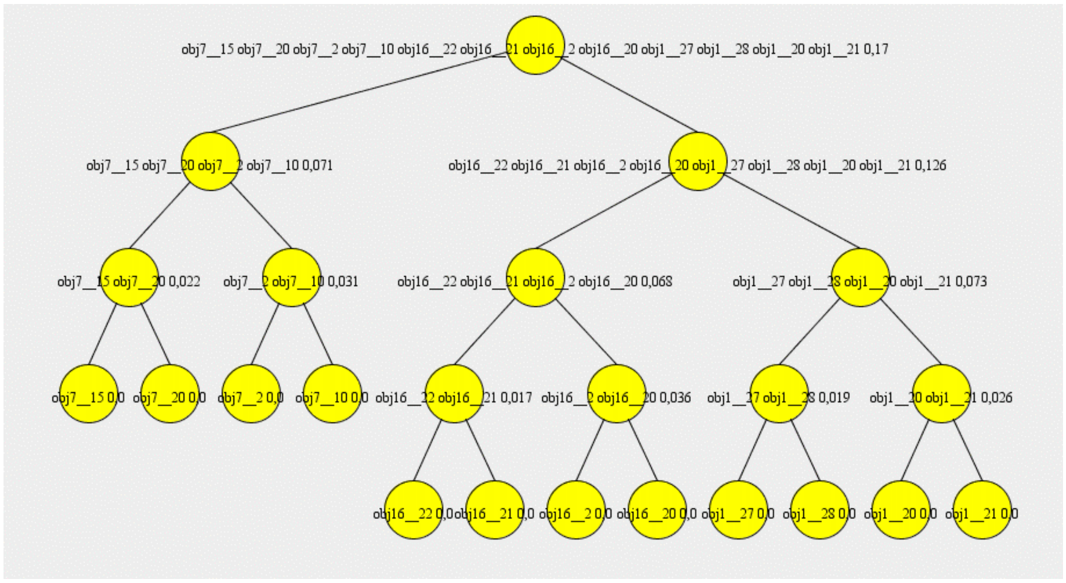

obj1__27 obj1__20 obj1__21 obj7__15 obj7__2 obj16__22 obj1__28 obj7__20 obj16__2 obj16__21 obj7__10 obj16__20 0,777 0
obj16__20 obj16__22 obj16__21 0,593 60
obj1__27 obj1__28 obj1__20 obj1__21 obj7__2 obj7__15 obj7__20 obj16__2 obj7__10 0,757 0
obj1__27 obj1__28 obj1__20 obj1__21 0,682 0
obj7__2 obj7__15 obj7__20 obj16__2 obj7__10 0,724 0
obj16__20 obj16__22 obj16__21 0,51 16

(c) Average pair group


obj16__2 obj16__20 obj16__22 obj16__21 obj7__15 obj7__2 obj7__20 obj7__10 obj1__27 obj1__28 obj1__20 obj1__21 0,691
obj16__2 obj16__20 obj16__22 obj16__21 0,594
obj7__15 obj7__2 obj7__20 obj7__10 obj1__27 obj1__28 obj1__20 obj1__21 0,666
obj16__2 0,0
obj16__20 obj16__22 obj16__21 0,522
obj7__15 obj7__2 obj7__20 0,568
obj7__10 obj1__27 obj1__28 obj1__20 obj1__21 0,626
obj16__20 0,0
obj16__22 obj16__21 0,471
obj7__15 0,0
obj7__2 obj7__20 0,533
obj7__10 obj1__27 obj1__28 0,585
obj1__20 obj1__21 0,588
obj16__22 0,0
obj16__21 0,0
obj7__2 0,0
obj7__20 0,0
obj7__10 0,0
obj1__27 obj1__28 0,518
obj1__20 0,0
obj1__21 0,0
obj1__27 0,0
obj1__28 0,0

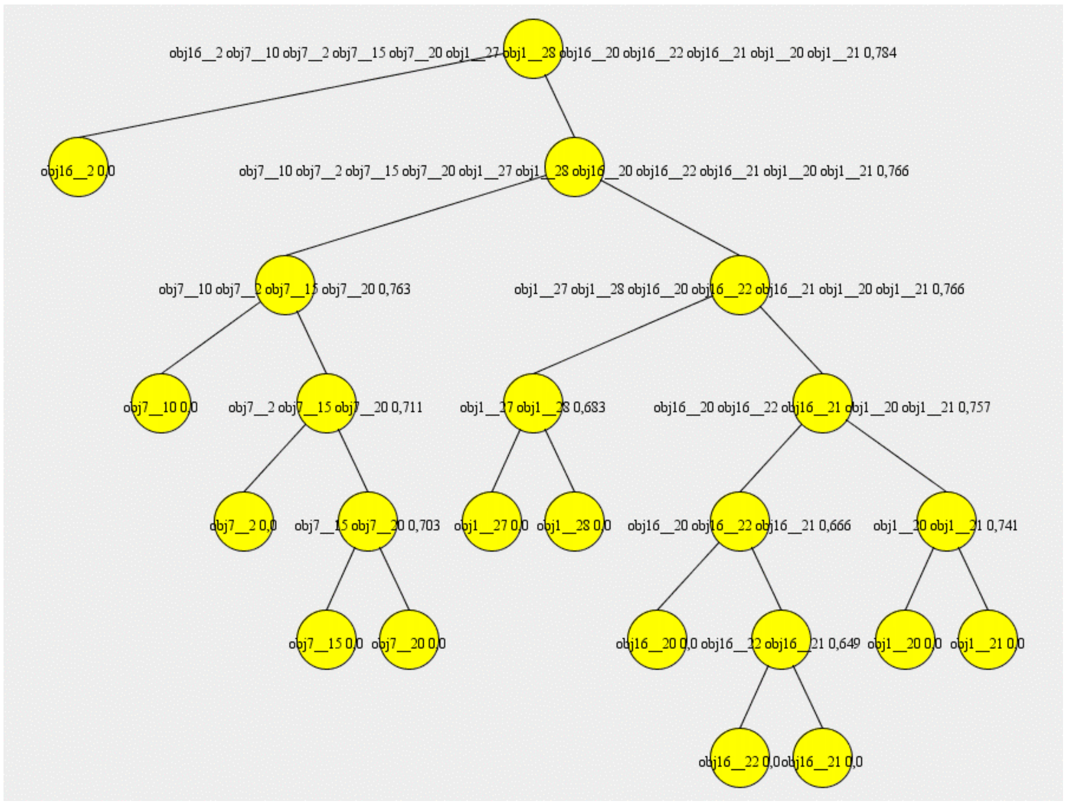
(d) Average within group

Figure C.30: Output of coil set with cosine and different linkages (cont.)


obj7__15 obj7__20 obj7__2 obj7__10 obj16__22 obj16__21 obj16__2 obj16__20 obj1__27 obj1__28 obj1__20 obj1__21 0,17
obj7__15 obj7__20 obj7__2 obj7__10 0,071
obj16__22 obj16__21 obj16__2 obj16__20 obj1__27 obj1__28 obj1__20 obj1__21 0,126
obj7__15 obj7__20 0,022
obj7__2 obj7__10 0,031
obj16__22 obj16__21 obj16__2 obj16__20 0,068
obj1__27 obj1__28 obj1__20 obj1__21 0,073
obj7__15 0,0
obj7__20 0,0
obj7__2 0,0
obj7__10 0,0
obj16__22 obj16__21 0,017
obj16__2 obj16__20 0,036
obj1__27 obj1__28 0,019
obj1__20 obj1__21 0,026
obj16__22 0,0
obj16__21 0,0
obj16__2 0,0
obj16__20 0,0
obj1__27 0,0
obj1__28 0,0
obj1__20 0,0
obj1__21 0,0

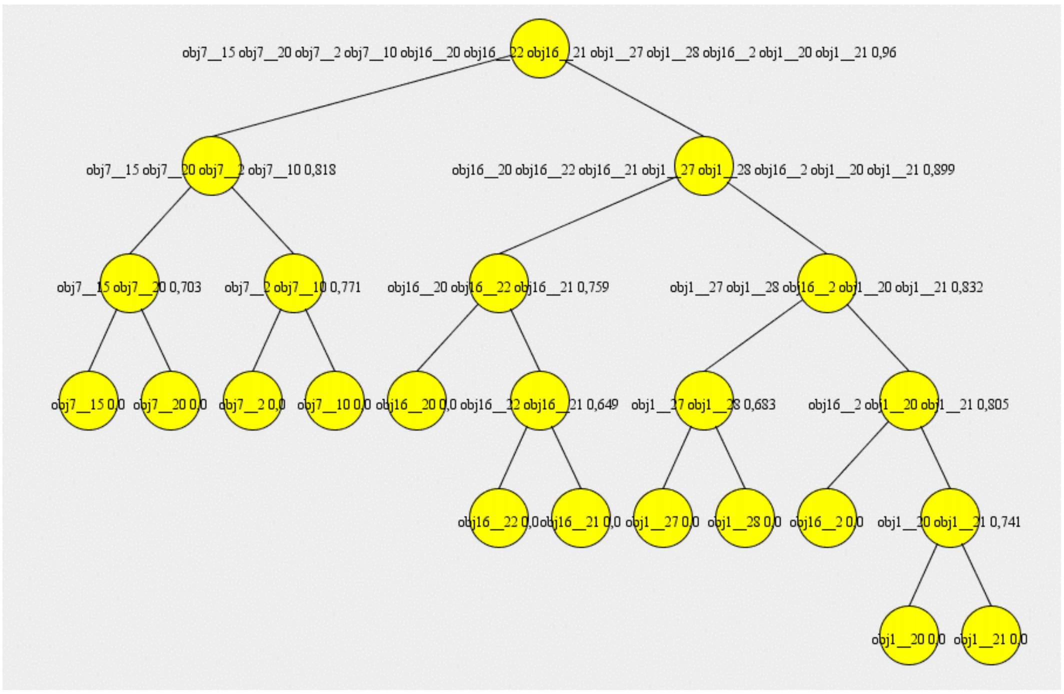
(e) Ward

Figure C.30: Output of coil set with cosine and different linkages (cont.)

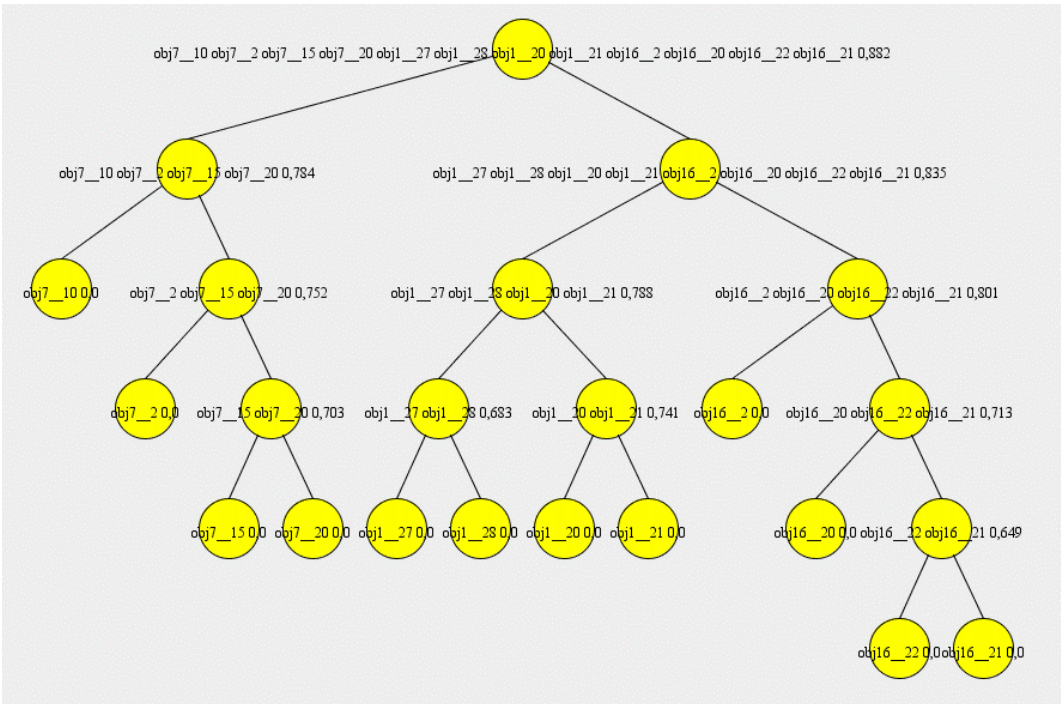

obj16__2 obj7__10 obj7__2 obj7__15 obj7__20 obj1__27 obj1__28 obj16__20 obj16__22 obj16__21 obj1__20 obj1__21 0,784
obj16__2 0,0
obj7__10 obj7__2 obj7__15 obj7__20 obj1__27 obj1__28 obj16__20 obj16__22 obj16__21 obj1__20 obj1__21 0,766
obj7__10 obj7__2 obj7__15 obj7__20 0,763
obj1__27 obj1__28 obj16__20 obj16__22 obj16__21 obj1__20 obj1__21 0,766
obj7__10 0,0
obj7__2 obj7__15 obj7__20 0,711
obj1__27 obj1__28 0,683
obj16__20 obj16__22 obj16__21 obj1__20 obj1__21 0,757
obj7__2 0,0
obj7__15 obj7__20 0,703
obj1__27 0,0
obj1__28 0,0
obj16__20 obj16__22 obj16__21 0,666
obj1__20 obj1__21 0,741
obj7__15 0,0
obj7__20 0,0
obj16__20 0,0
obj16__22 obj16__21 0,649
obj1__20 0,0
obj1__21 0,0
obj16__22 0,0
obj16__21 0,0

(a) Single

Figure C.31: Output of coil with Tanimoto and different linkages

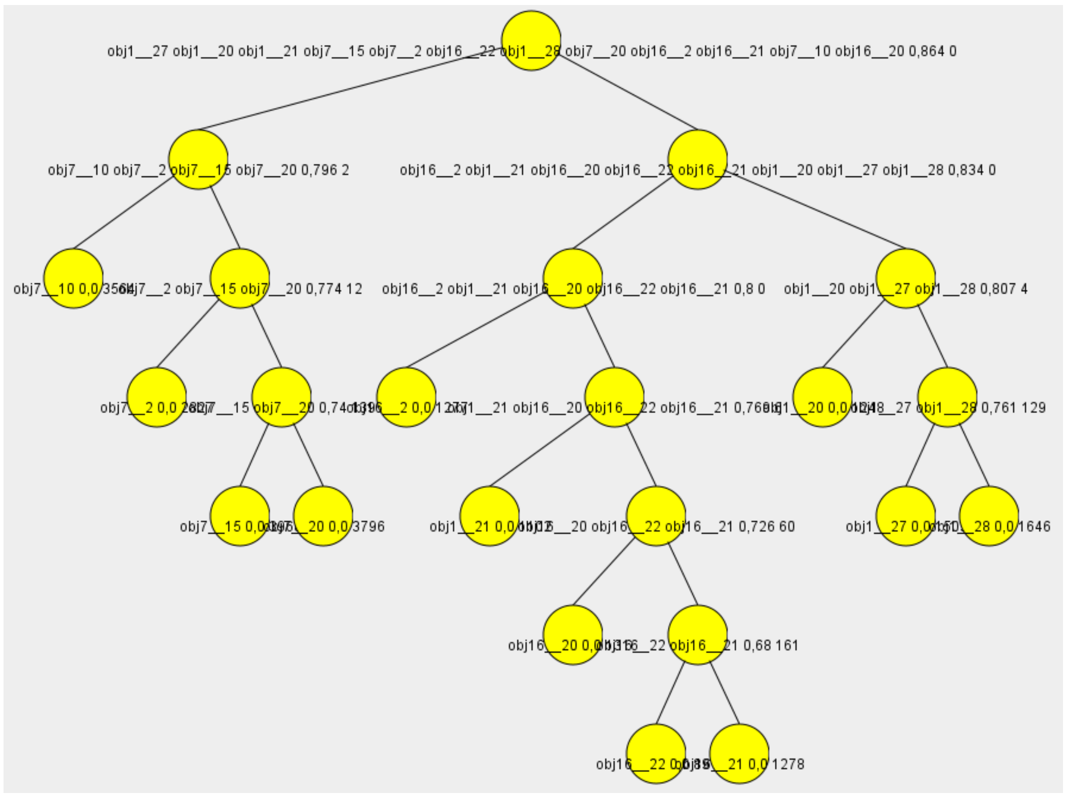

obj7__15 obj7__20 obj7__2 obj7__10 obj16__20 obj16__22 obj16__21 obj1__27 obj1__28 obj16__2 obj1__20 obj1__21 0,96
obj7__15 obj7__20 obj7__2 obj7__10 0,818
obj16__20 obj16__22 obj16__21 obj1__27 obj1__28 obj16__2 obj1__20 obj1__21 0,899
obj7__15 obj7__20 0,703
obj7__2 obj7__10 0,771
obj16__20 obj16__22 obj16__21 0,759
obj1__27 obj1__28 obj16__2 obj1__20 obj1__21 0,832
obj7__15 0,0
obj7__20 0,0
obj7__2 0,0
obj7__10 0,0
obj16__20 0,0
obj16__22 obj16__21 0,649
obj1__27 obj1__28 0,683
obj16__2 obj1__20 obj1__21 0,805
obj16__22 0,0
obj16__21 0,0
obj1__27 0,0
obj1__28 0,0
obj16__2 0,0
obj1__20 obj1__21 0,741
obj1__20 0,0
obj1__21 0,0

(b) Complete

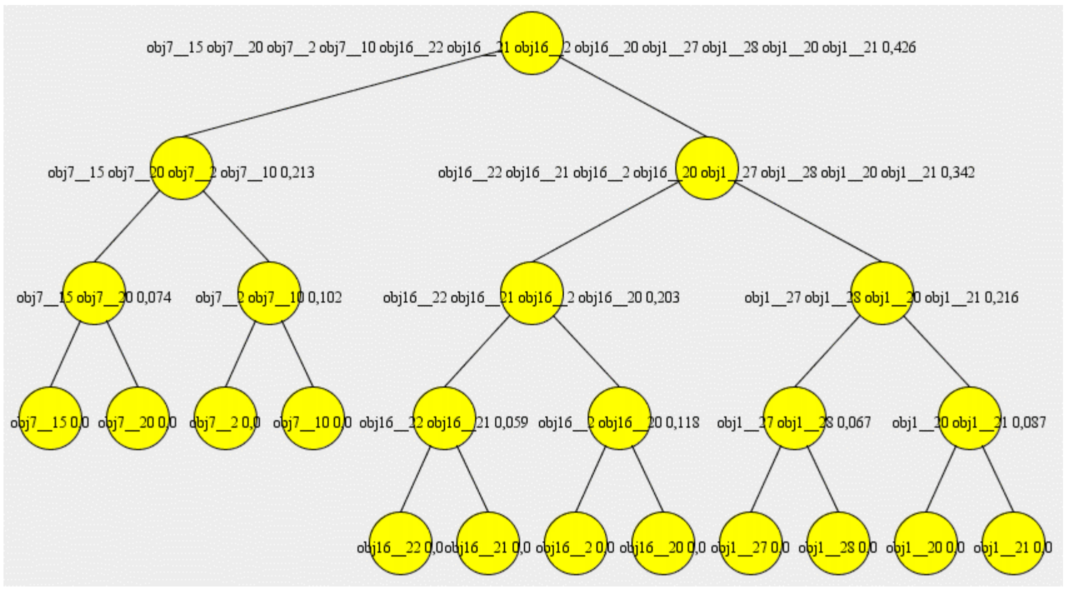

obj7__10 obj7__2 obj7__15 obj7__20 obj1__27 obj1__28 obj1__20 obj1__21 obj16__2 obj16__20 obj16__22 obj16__21 0,882
obj7__10 obj7__2 obj7__15 obj7__20 0,784
obj1__27 obj1__28 obj1__20 obj1__21 obj16__2 obj16__20 obj16__22 obj16__21 0,835
obj7__10 0,0
obj7__2 obj7__15 obj7__20 0,752
obj1__27 obj1__28 obj1__20 obj1__21 0,788
obj16__2 obj16__20 obj16__22 obj16__21 0,801
obj7__2 0,0
obj7__15 obj7__20 0,703
obj1__27 obj1__28 0,683
obj1__20 obj1__21 0,741
obj16__2 0,0
obj16__20 obj16__22 obj16__21 0,713
obj7__15 0,0
obj7__20 0,0
obj1__27 0,0
obj1__28 0,0
obj1__20 0,0
obj1__21 0,0
obj16__20 0,0
obj16__22 obj16__21 0,649
obj16__22 0,0
obj16__21 0,0

(c) Average pair group

Figure C.31: Output of coil with Tanimoto and different linkages (cont.)

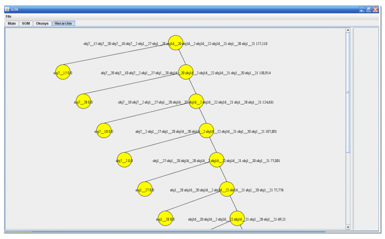

obj1__27 obj1__20 obj1__21 obj7__15 obj7__2 obj16__22 obj1__28 obj7__20 obj16__2 obj16__21 obj7__10 obj16__20 0,864 0
obj7__10 obj7__2 obj7__15 obj7__20 0,796 2
obj16__2 obj1__21 obj16__20 obj16__22 obj16__21 obj1__20 obj1__27 obj1__28 0,834 0
obj16__2 obj1__21 obj16__20 obj16__22 obj16__21 0,8 0
obj1__20 obj1__27 obj1__28 0,807 4
obj1__28 0,761 129
obj16__22 obj16__21 0,726 60
obj16__22 obj16__21 0,68 161

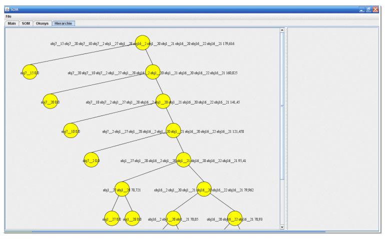
(d) Average within group

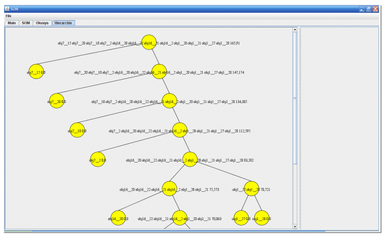

obj7__15 obj7__20 obj7__2 obj7__10 obj16__22 obj16__21 obj16__2 obj16__20 obj1__27 obj1__28 obj1__20 obj1__21 0,426
obj7__15 obj7__20 obj7__2 obj7__10 0,213
obj16__22 obj16__21 obj16__2 obj16__20 obj1__27 obj1__28 obj1__20 obj1__21 0,342
obj7__15 obj7__20 0,074
obj7__2 obj7__10 0,102
obj16__22 obj16__21 obj16__2 obj16__20 0,203
obj1__27 obj1__28 obj1__20 obj1__21 0,216
obj7__15 0,0
obj7__20 0,0
obj7__2 0,0
obj7__10 0,0
obj16__22 obj16__21 0,059
obj16__2 obj16__20 0,118
obj1__27 obj1__28 0,067
obj1__20 obj1__21 0,087
obj16__22 0,0
obj16__21 0,0
obj16__2 0,0
obj16__20 0,0
obj1__27 0,0
obj1__28 0,0
obj1__20 0,0
obj1__21 0,0

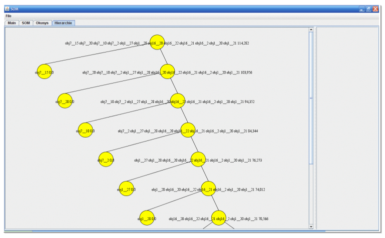
(e) Ward

Figure C.31: Output of coil with Tanimoto and different linkages (cont.)


(a) Single


(b) Complete


(c) Average pair group


(d) Average within group

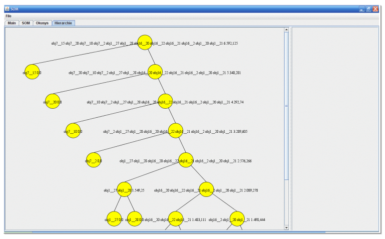

(e) Ward

Figure C.32: Output of coil with Euclidian and different linkages

List of Figures

List of Tables

Bibliography

Elke Achert. *Hierarchical Subspace Clustering.* Phdthesis, Ludwig-Maximilians-University, Munich, April 2007.

James A. Anderson, Andras Pellionisz, and Edward Rosenfeld. *Neurocomputing (vol. 2): directions for research.* MIT Press, Cambridge, MA, USA, 1990. ISBN 0-262-01119-0.

Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 49–60. ACM Press, 1999. ISBN 1-58113-084-8.

C. M Benjamin, Ke Fung, and Martin Ester Wang. Large hierarchical document clustering using frequent itemsets. In *In Proc. SIAM International Conference on Data Mining 2003 (SDM 2003*, 2003.

Pavel Berkhin. Survey of clustering data mining techniques. Technical report, 2002.

James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 1981. ISBN 0306406713.

G. Brunner and H. Burkhardt. Structure features for content-based image retrieval. In *Proceedings of the 27th DAGM Symposium*, number 3663, Wien, 2005. Springer.

Gerd Brunner. *Structure Features for Content-Based Image Retrieval and Classification Problems.* PhD thesis, Albert-Ludwigs-Universität Freiburg, 2006.

Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, New York, NY, USA, 2002. ACM. ISBN 1-58113-495-9. doi: http://doi.acm.org/10.1145/509907.509965.

Kenneth L. Clarkson. Nearest-neighbor searching and metric space dimensions. In *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, 2006.

Joaquim P. Marques de Sá. *Pattern Recognition - Concepts, Methods and Applications*. Springer, Berlin, 2001.

D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977. doi: 10.1093/comjnl/20.4.364. URL http://comjnl.oxfordjournals.org/cgi/content/abstract/20/4/364.

A.P. Dempster. A generalization of bayesian inference. Technical report, HARVARD UNIV CAMBRIDGE MASS DEPT OF STATISTICS, 1967.

Michel M. Deza and Monique Laurent. *Geometry of Cuts and Metrics*, volume 15 of *Algorithms and Combinatorics*. Springer, 1997.

Michael Dittenbach, Dieter Merkl, and Andreas Rauber. Hierarchical clustering of document archives with the growing hierarchical self-organizing map. In *Artificial Neural Networks — ICANN 2001*, pages 500–505. 2001. URL http://www.springerlink.com/content/ptj02a1e54dmhfng.

Didier Dubois and Henri Prade. *Fuzzy sets and systems: Theory and applications*, volume 144 of *Mathematics in Science and Engineering*. Academic Press, New York, 1980.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, New York, 2nd edition, 2001.

J.C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, 67:47–55, 1992.

James S. Farris. On the cophenetic correlation coefficient. *Systematic Zoology*, 18(3):279–285, 1969. ISSN 00397989. doi: http://dx.doi.org/10.2307/2412324. URL http://dx.doi.org/10.2307/2412324.

Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Mach. Learn.*, 2(2):139–172, September 1987. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1022852608280. URL http://dx.doi.org/10.1023/A:1022852608280.

J.A. Flanagan. Unsupervised clustering of symbol strings. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 4, pages 3250– 3255, 2003.

John A. Flanagan. Self-organisation in kohonen's som. *Neural Netw.*, 9(7):1185–1197, 1996. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/0893-6080(96)00038-X.

John A. Flanagan and Johan Himberg. A hierarchical approach to learning context and facilitating user interaction. In *in Mobile Devices, in Artificial Intelligence in Mobile System 2003 (AIMS 2003) (in conjunction with Ubicomp 2003), October 12*, pages 249–254, 2003.

J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artif. Intell.*, 40(1-3):11–61, 1989. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/0004-3702(89)90046-5.

J. Ghosh and A. Strehl. Similarity-based text clustering: A comparative study. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, *Grouping Multidimensional Data*, pages 73–97. Springer, Berlin, Heidelberg, 2006. doi: 10.1007/3-540-28349-8.

K. Chidananda Gowda and E. Diday. Symbolic clustering using a new dissimilarity measure. *Pattern Recogn.*, 24(6):567–578, 1991. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/0031-3203(91)90022-W.

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. pages 73–84, 1998. URL citeseer.ist.psu.edu/guha98cure.html.

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. *Data Engineering, International Conference on*, 0:512, 1999. ISSN 1063-6382. doi: http://doi.ieeecomputersociety.org/10.1109/ICDE.1999.754967.

S.S. Hacisalihzade, L.W. Stark, and J.S. Allen. Visual perception and sequences of eye movement fixations: astochastic modeling approach. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3), 1992.

J. Herrero and J. Dopazo. Combining hierarchical clustering and self-organizing maps for exploratory analysis of gene expression patterns. *Journal of Proteome Research*, 1(5):467–470, 2002. ISSN 1535-3893. URL http://pubs3.acs.org/acs/journals/doilookup?in_doi=10.1021/pr025521v.

A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, 1988.

A.K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/331499.331504.

Jari Kangas. *On the Analysis of Pattern Sequences by Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, 1994. URL http://nucleus.hut.fi/~jari/papers/thesis94.ps.Z.

George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/2.781637.

L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

Jens Keuchel, Matthias Heiler, and Christoph Schnörr. Hierarchical image segmentation based on semidefinite programming. In *Pattern Recognition*, volume 3175, pages 120–128. Springer, 2004.

M. Kikuchi and K. Fukushima. Pattern recognition with eye movement: a neural network model. *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, 2:37–40 vol.2, 2000. doi: 10.1109/IJCNN.2000.857871.

E. I. Knudsen, S. du Lac, and S. D. Esterly. Computational maps in the brain. *Annu Rev Neurosci*, 10:41–65, 1987. ISSN 0147-006X. doi: http://dx.doi.org/10.1146/annurev.ne.10.030187.000353. URL http://dx.doi.org/10.1146/annurev.ne.10.030187.000353.

Teuvo Kohonen. *Self-organizing maps*, volume 30 of *Springer series in information sciences*. Springer, Berlin, Heidelberg, New York, 3rd edition, 2001.

Teuvo Kohonen. Analysis of a simple self-organizing process. *Biological Cybernetics*, 44(2):135–140, 1982a. doi: 10.1007/BF00317973.

Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, January 1982b. doi: 10.1007/BF00337288.

Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. doi: 10.1109/5.58325. URL http://dx.doi.org/10.1109/5.58325.

Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21:1–6, November 1998.

S. Krishnamachari and M. Abdel-Mottaleb. Hierarchical clustering algorithm for fast image retrieval. In *IS&T/SPIE Conference on Storage and Retrieval for Image and Video databases VII*.

Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, New York, NY, USA, 1999. ACM. ISBN 1-58113-143-7. doi: http://doi.acm.org/10.1145/312129.312186.

Risto Miikkulainen. Script recognition with hierarchical feature maps. *Connection Science*, 2:83–101, 1990.

Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia object image library (coil-20). Technical report cucs-005-96, Columbia University, Department of Computer Science, New York, February 1996. URL http://www.cs.columbia.edu/CAVE/databases/papers/nene/nene-nayar-murase-coil-20.ps.

David Noton and Lawrence Stark. Scanpaths in Eye Movements during Pattern Perception. *Science*, 171(3968):308–311, 1971. doi: 10.1126/science.171.3968.308. URL http://www.sciencemag.org/cgi/content/abstract/171/3968/308.

Noboru Ohsumi and Hirokazu Moroi. A note on the evaluation of the agglomerative hierarchical clustering methods. *Behaviormetrika*, 2(2):61–72, 1975. ISSN 1349-6964. doi: 10.2333/bhmk.2.61. URL http://www.journalarchive.jst.go.jp/jnlpdf.php?cdjournal=bhmk1974\&cdvol=2\&noissue=2\&startpage=61\&lang=en\&from=jnlabstract.

Jörg Ontrup and Helge Ritter. Text categorization and semantic browsing with self-organizing maps on non-euclidean spaces. In *PKDD '01: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 338–349, London, UK, 2001. Springer-Verlag. ISBN 3-540-42534-9.

Witold Pedrycz. *Knowledge-Based Clustering: From Data to Information Granules*. Wiley & Sons, 1st edition, February 2005.

Thomas Reineking. Active vision-based localization using dempster-shafer theory. Master's thesis, University of Bremen, Faculty of Computer Science and Mathematics, Bremen, May 2008.

Raul Rojas. *Theorie der neuronalen Netze*. Springer, Berlin, 1996.

A Roucoux, D Guitton, and M Crommelinck. Stimulation of the superior colliculus in the alert cat. ii. eye and head movements evoked when the head is unrestrained. *Experimental brain research*, 39(1):75–85, April 1980.

I. Rybak, V. Gusakova, A. Golovan, L. Podladchikova, and N. Shevtsova. A model of attention-guided visual perception and recognition. *Vision Research*, 38(15-16):2387–2400, August 1998. doi: http://dx.doi.org/10.1016/S0042-6989(98)00020-0. URL http://dx.doi.org/10.1016/S0042-6989(98)00020-0.

I. A. Rybak, V. I. Gusakova, A. V. Golovan, L. N. Podladchikova, , and N. A. Shevtsova. Attention-guided recognition based on "what" and "where" representations: A behavioral model. pages 663–670. Elsevier, 2005.

Albert Ali Salah, Ethem Alpaydin, and Lale Akarun. A selective attention-based method for visual pattern recognition with application to handwritten digit recognition and face recognition. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 24(3), 2002.

Kerstin Schill. Decision support systems with adaptive reasoning strategies. In *Foundations of Computer Science: Potential - Theory - Cognition, to Wilfried Brauer on the occasion of his sixtieth birthday*, pages 417–427, London, UK, 1997. Springer-Verlag. ISBN 3-540-63746-X.

Kerstin Schill, Elisabeth Umkehrer, Stephan Beinlich, Gerhard Krieger, and Christoph Zetzsche. Scene analysis with saccadic eye movements: Top-down and bottom-up modeling. *Journal of Electronic Imaging*, 10(1):152–160, January 2001.

Glenn Shafer. *A mathematical theory of evidence*. Princeton University Press, Princeton, N.J., 1976.

R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973. URL http://comjnl.oxfordjournals.org/cgi/content/abstract/16/1/30.

R. R. Sokal and R. J. Rohlf. The comparisons of dendrograms by objective methods. *Taxon*, 11:33–40, 1962.

Helmuth Spaeth. *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*. Ellis Horwood, 1980.

M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques, 2000. URL http://citeseer.ist.psu.edu/steinbach00comparison.html.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas. Som toolbox for matlab 5. Technical report, Helsinki University of Technology, 2000. URL http://www.cis.hut.fi/projects/somtoolbox/package/papers/techrep.pdf.

Ellen M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. Technical report, Ithaca, NY, USA, 1986.

Junbai Wang, Jan Delabie, Hans Aasheim, Erlend Smeland, and Ola Myklebost. Clustering of the som easily reveals distinct gene expression patterns: results of a reanalysis of lymphoma study. *BMC Bioinformatics*, 3(1):36, 2002. ISSN 1471-2105. doi: 10.1186/1471-2105-3-36. URL http://www.biomedcentral.com/1471-2105/3/36.

Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, March 1963. URL http://www.jstor.org/stable/2282967?seq=1.

Satosi Watanabe. *Knowing and Guessing: A Quantitative Study of Inference and Information*. Wiley, New York, 1969.

Ian H. Witten and Eibe Frank. *Data Mining. Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2nd edition, 2005.

Johannes Wolter. Entwicklung eines sensomotorischen explorationssystems zur klassifikation von vr-umgebungen. Master's thesis, University of Bremen, Faculty of Computer Science and Mathematics, Bremen, April 2006.

R H Wurtz and J E Albano. Visual-motor function of the primate superior colliculus. *Annual Review of Neuroscience*, 3(1):189–226, 1980. doi: 10.1146/annurev.ne.03.030180.001201. URL http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.ne.03.030180.001201. PMID: 6774653.

H. Xu and D.K. Agrafiotis. Nearest neighbor search in general metric spaces using a tree data structure with a simple heuristic. *Journal of Chemical Information and Computer Sciences*, 43(6):1933–1941, 2003. ISSN 1549-9596. URL http://pubs3.acs.org/acs/journals/doilookup?in_doi=10.1021/ci034150f.

Méziane Yacoub, Fouad Badran, and Sylvie Thiria. A topological hierarchical clustering: Application to ocean color classification. In *ICANN '01: Proceedings of the International Conference on Artificial Neural Networks*, pages 492–499, London, UK, 2001. Springer-Verlag. ISBN 3-540-42486-5.

Lofti A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965. URL http://www-bisc.cs.berkeley.edu/Zadeh-1965.pdf.

Christoph Zetzsche and Gerhard Krieger. Intrinsic dimensionality: nonlinear image operators and higher-order statistics. pages 403–441, 2001.

Christoph Zetzsche and Ulrich Nuding. Nonlinear and higher-order approaches to the encoding of natural scenes. *Network: Computation in Neural Systems*, 16(2-3):191–221, June/September 2005.

Christoph Zetzsche, Kerstin Schill, H. Deubel, G. Krieger, Elisabeth Umkehrer, and S. Beinlich. Investigation of a sensorimotor system for saccadic scene analysis: an integrated approach. In *Proceedings of the fifth international conference on simulation of adaptive behavior on From animals to animats 5*, pages 120–126, Cambridge, MA, USA, 1998. MIT Press. ISBN 0-262-66144-6.

Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer, Berlin, 2005.

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 103–114. ACM Press, 1996.

Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. Technical report, Department of Computer Science, University of Minnesota., Minnesota, 2001.

Statutory Declaration

I declare that I have developed and written the enclosed thesis entirely by myself and have not used sources or means without declaration in the text.

Any thoughts or quotations which were inferred from these sources are clearly marked as such.

Bremen, November 10, 2008

Konrad Gadzicki