Ersch. in: Advances in Fuzzy Clustering and its Applications / José Valente de Oliveira, Witold Pedrycz (eds.). - Chichester : Wiley, 2007. - S. 123-136. - ISBN 978-0-470-02760-8

6

Interactive Exploration of Fuzzy Clusters

Bernd Wiswedel[1], David E. Patterson[2], and Michael R. Berthold[1]

[1]*Department of Computer and Information Science, University of Konstanz, Germany*
[2]*Vistamount Consulting, USA*

6.1 INTRODUCTION

Classical learning algorithms create models of data in an uncontrolled, non-interactive manner. Typically the user specifies some (method-dependent) parameters like distance function or number of clusters that he/she likes to identify, followed by the application of the algorithm using these settings. The process of the model generation itself, however, cannot be controlled or influenced by the user. The final outcome is then evaluated by means of some quality measure, for instance the classification error for supervised learning or some cluster validity measure for unsupervised tasks, or it is judged based on the user's impression, provided that the model is interpretable. Depending on the quality of the results, the model generation is either considered successful, which means the model is a good summarization of the data and can be used, for example, for further classification tasks, or it requires further fine-tuning of the parameters and a rerun of the algorithm.

This "learning" scheme is characteristic for most methods in machine learning and data mining. However, in many applications the focus of analysis is not on the optimization of some quality function but rather on the user-controlled generation of interpretable models in order to incorporate background knowledge. This requires tools that allow users to interact with the learning algorithm to inject domain knowledge or help to construct a model manually by proposing good model components, for example, cluster prototypes or classification rules, which can then be accepted, discarded, or fine-tuned. There are generally two different extrema of cooperation between user and system: either the user or the system has all the control and carries out the model generation; however, in general it requires a balance of both.

Several approaches to build interpretable models interactively for classification have been proposed in the past. Ankerst, Elsen, Ester, and Kriegel (1999) propose an interactive decision tree learner that is based on a pixel-oriented visualization of the data. Figure 6.1 shows a screenshot of such an interactive tool. The visualization technique is similar to the one proposed by Ankerst, Elsen, Ester, and Kriegel (1999) but uses a box instead of a circle view. The top half of the figure displays the class distribution for one of the input dimensions with each pixel representing one of the training objects (there are about 15000 training objects

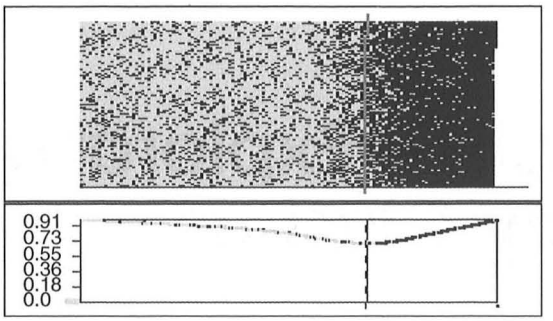


Figure 6.1 Interactive construction of decision trees.

in this data-set). The objects are aligned column by column according to the order of their values in the current dimension, whereby the object with the smallest value is plotted at the top left corner of the pixel display and the others are lined using a snake like alignment. Different colors represent different classes, for instance there are two classes in the example in Figure 6.1. The bottom half of the figure shows the entropy curve for the current dimension, i.e., a classical (automatic) decision tree learner would perform a split at the value with the smallest entropy over all dimensions. This visualization technique allows the class distribution to be inspected immediately for each individual dimension, either by plotting different boxes for different classes or – as proposed by Ankers, Elsen, Ester, and Kriegel (1999) – by using a circle view where different segments of the circle are used to plot the distribution in different dimensions. The construction of the decision tree model will take place interactively whereby the system supports the user by proposing good split points, allowing for look-aheads (such as what would be a resulting sub tree if a split was performed), or automatic construction of sub-trees, for example, when there are only small amounts of data left in a decision branch. The user, on the other hand, can adjust split points or also perform the split on another attribute, which may not lead to such a high information gain as the numerical optimal split point but is, from the user's perspective, more meaningful.

This system for decision tree construction is a typical example for an interactive model generation tool. By using an appropriate visualization technique (in the above example a pixel display), the user can guide the model generation.

Decision trees always partition the instance space, i.e., each point in the input space is assigned a class label. This is often not desirable as some regions of the input space may not contain (training-) data at all and therefore there is no obvious evidence to prefer one class over the other. Rule learning algorithms, on the other hand, generate a set of rules whereby each rule covers only a relatively small region of the input data. Typically, there may be regions in the input space for which more than one rule fires (posing the challenge of conflict handling when the firing rules are of contradicting classes) or for which none of the rules is active. In the latter case, an outcome is often determined using the majority class (the one with highest a priori probability) or – and which is often preferable – a "don't know" answer. Many rule learning algorithms also use the notion of fuzzy membership functions to model regions of high and low confidence (Berthold, 2003; Chiu, 1997).

Fuzzy clusters, similar to fuzzy rules, are well suited for presentation of the resulting classification model to the user. Although the traditional cluster algorithm works on unsupervised data-sets, extensions also allow cluster models to be built that distinguish between areas of different classes. This is an intriguing approach especially for cases where one expects to find various, distributed areas that belong to the same class. Often these clusters are then used directly as fuzzy rules or serve to initialize a fuzzy rule system, which is then optimized. A typical example of this sort of algorithm has been proposed by Chiu (1997): it first finds a set of clusters for each class using subtractive clustering (Chiu, 1994), an algorithm that builds upon the well-known mountain method by Yager and Filev (1994), and then derives classification rules from them.

In this chapter we focus on a supervised approach to construct a set of fuzzy clusters for classification. The algorithm does not use a two stage learning such as in (Chiu, 1997) but rather generates potentially

discriminative fuzzy clusters from the beginning. It initially constructs a so-called *Neighborgram* for each object of interest. A Neighborgram is a summarization of the neighborhood of an object, which allows an interpretable view on the underlying data. Such a complete and hence computationally expensive approach obviously only works for all classes of a medium size data-set or – in the case of very large data-sets – to model a minority class of interest. However, in many applications the focus of analysis is on a class with few objects only, a minority class. Such data can be found, for instance, in drug discovery research. Here, huge amounts of data are generated in high throughput screening, but only very few compounds really are of interest to the biochemist. Therefore, it is of prime interest to find clusters that model a small but interesting subset of data extremely well.

The algorithm finds clusters in a set of such Neighborgrams based on an optimality criterion. Since Neighborgrams are easy to interpret, the algorithm can also be used to suggest clusters visually to the user, who is able to interact with the clustering process in order to inject expert knowledge. Therefore, the clustering can be performed fully automatically, interactively, or even completely manually. Furthermore, constructing Neighborgrams only requires a distance matrix, which makes them applicable to data-sets where only distances between objects are known. For many similarity metrics in molecular biology no underlying feature values are known since those similarity (and hence also the distance) values are computed directly. In contrast, methods that compute cluster representatives as mixtures of training objects (like fuzzy c-means by Bezdek (1981)) do require the availability of an underlying feature representation in order to continuously compute and update the cluster centers.

Neighborgrams can naturally be applied to problem settings where there are multiple descriptors for the data available, known as *parallel universes*. One such application is biological data analysis where different descriptors for molecules exist but none of them by itself shows global satisfactory prediction results. We will demonstrate how the Neighborgram clustering algorithm can be used to exploit the information of having different descriptors and how it finds clusters spread out of different universes, each modeling a small subset of the data.

This chapter is organized as follows. We first introduce the concept of Neighborgrams and describe the basic clustering algorithm. We then extend the algorithm to also handle fuzzy clusters before Section 6.3 discusses some aspects of the visual exploration and demonstrates the usefulness of the visual clustering procedure. In Section 6.4 we focus on learning in parallel universes and give an example application using the Neighborgram algorithm.

6.2 NEIGHBORGRAM CLUSTERING

This section introduces Neighborgrams as an underlying data structure of the presented algorithm. We will formalize this structure and derive some properties which help us to judge the quality of a Neighborgram later on. We will refer to one of our previous articles, which discusses the automatic classifier using Neighborgrams more extensively (Berthold, Wiswedel, and Patterson, 2005).

We will assume a set of training objects T with $|T| = M$ instances for which distances, $d(x_i, x_j), i, j \in \{1, \ldots, M\}$, are given[1]. Each example is assigned to one of C classes, $c(x_i) = k, 1 \leq k \leq C$.

6.2.1 Neighborgrams

A Neighborgram is a one-dimensional model of the neighborhood of a chosen object, which we will call the *centroid*. Other objects are mapped into the Neighborgram depending on their distance to this centroid. Essentially, a Neighborgram summarizes the neighborhood of the centroid through a ray on

[1]Note that it is not necessary to know the feature values for an instance. It is sufficient to provide the algorithm with distances between objects.

to which the closest neighbors of the centroid are plotted. Obviously, mapping all objects on to the ray would be complicated and the visualization would lose its clarity. Therefore, we introduce a parameter R that determines the maximum number of objects stored in a Neighborgram. Those R stored items represent the R-closest neighbors to the centroid. Hence, a Neighborgram for a certain centroid x_i can also be seen as an ordered list of length R:

$$\mathrm{NG}_i = [x_{l_1}, \ldots, x_{l_R}].$$

The list NG_i is sorted according to the distance of object x_{l_r} to the center vector x_i:

$$\forall r \ : \ 2 \leq r \leq R \wedge d(x_i, x_{l_{(r-1)}}) \leq d(x_i, x_{l_r}),$$

and the objects in the Neighborgram are the closest neighbors of the centroid:

$$\neg \exists r \ : \ r > R \wedge d(x_i, x_{l_r}) < d(x_i, x_{l_R}).$$

Note that $l_1 = i$, because $d(x_i, x_i) = 0$ for all i, that is, each object is closest to itself. Note also that this list is not necessarily a unique representation since it is possible that two entries in the list have exactly the same distance to the centroid. The order of those items would then not be uniquely defined. However, as we will see later, this does not affect the outcome of the clustering algorithm that we are discussing here.

Obviously in the case of large data-sets the computation of Neighborgrams for each training object is excessively time and memory consuming. However, as noted earlier, the main target of the algorithm discussed here are problems where one (or several) minority class(es) are of prime interest. The computation of Neighborgrams for all these objects is then of complexity $O(R \cdot M \cdot M')$, where M' indicates the number of examples of the minority class(es), i.e., $M' \ll M$ in the case of large data-sets. This complexity estimate is derived as follows: for each object ($O(M)$) and for each Neighborgram ($O(M')$) do an insertion sort into a list of R objects ($O(R)$). If the size R of the Neighborgrams is closer to the overall number of objects M it might make more sense to use a more efficient sorting scheme but for large data-sets usually $R \ll M$ holds and an insertion sort is sufficiently efficient. For large data-sets, M will dominate the above estimate and result in a roughly linear complexity.

Figure 6.2 shows an example of two Neighborgrams for the famous Iris data (Fisher, 1936), a data-set containing four-dimensional descriptions of three different types of Iris plant. For the sake of simplicity, in the example we use only two of the original four dimensions in order to be able to display them in a scatterplot (left-hand side). Different colors are used here to show the different class memberships of the overall 150 objects. Two sample Neighborgrams are shown on the right. They are constructed for the two centroids of the gray class (Iris-Setosa), indicated by the arrows in the scatterplot. Each Neighborgram summarizes the neighborhood of its centroid. The centroid of the top Neighborgram, for instance, has many objects from the same (gray) class in its close vicinity as can be immediately seen when looking at the Neighborgram. The visualization technique uses a simple ray to plot neighbors; however, whenever two objects are too close to each other and might overlap, we stack them. Stacking allows the user to individually select certain objects in one Neighborgram, which are then also highlighted in the other Neighborgrams or another system.

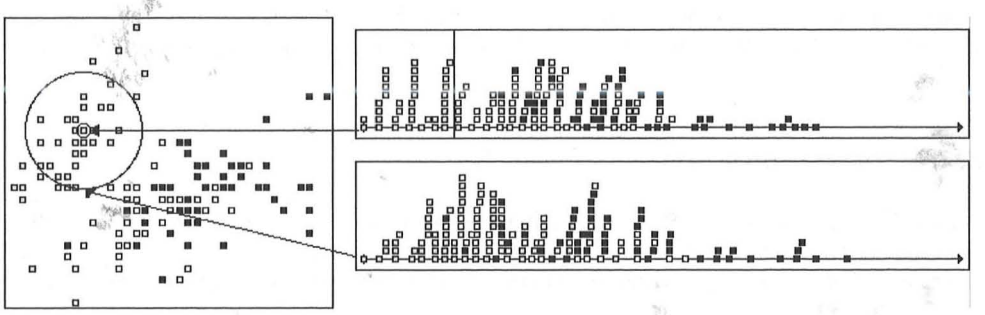


Figure 6.2 Two Neighborgrams for the Iris data is shown on the left. The two-dimensional input space, on the right there are two different Neighborgrams for two selected objects displayed.

Neighborgrams are constructed for all objects that are of interest to the user, for instance all objects of one class. Just from looking at the example in Figure 6.2, we can qualitatively rank Neighborgrams: the centroid of the top Neighborgram has many objects of its class in the close neighborhood whereas the centroid of the bottom Neighborgram is surrounded by some gray but also white objects. The top Neighborgram therefore suggests a better cluster candidate since new, unseen objects that have a small distance to the centroid are likely to be of the same class.

Let us briefly sketch the underlying algorithm to identify a set of good Neighborgrams in the next section before we derive some underlying properties of a Neighborgram using the notation introduced above.

6.2.2 The Basic Clustering Algorithm

The key idea underlying the clustering algorithm is that each object, for which a Neighborgram has been built, is regarded as a potential cluster center. The objective of the algorithm is to rank Neighborgrams in order to greedily find the "best" cluster at each step. The result is a subset of all possible clusters that covers a sufficient number of objects.
The algorithm can be summarized as follows:

1. determine a cluster candidate for each Neighborgram;
2. rank cluster candidates and add the best one as a cluster;
3. remove all objects covered by this cluster;
4. start over at Step 1, unless certain stopping criteria are fulfilled.

Obviously, it needs to be defined, what a cluster candidate is how these candidates can be ranked it, and what removing covered objects really means. In addition, the termination criterion has to be specified. In order to do this, let us first define a few properties of Neighborgrams.

6.2.3 Neighborgram Properties

In Section 6.2.1 we used an ordered list as representation for a Neighborgram. This list contains objects, which are ordered according to their distance to the centroid. The length of the list is determined by the parameter R:

$$\mathrm{NG}_i = [x_{l_1}, \ldots, x_{l_r}, \ldots, x_{l_R}].$$

The main parameters to describe a cluster candidate are the following:

- *Coverage* Γ. The default coverage of a cluster with a certain depth $r \leq R$ determines how many positive objects it "explains," that is, the number of objects of the same class as the centroid that fall within its radius:

$$\Gamma_i(r) = |\{x_{l_{r'}} \in \mathrm{NG}_i | 1 \leq r' \leq r \wedge c(x_{l_{r'}}) = c(x_i)\}|.$$

- *Purity* Π. The purity of a Neighborgram is the ratio of the number of objects belonging to the same class as the centroid to the number of objects encountered up to a certain depth $r \leq R$. The purity is a measure of how many positive vs. negative objects a certain neighborhood around the centroid contains. Positive objects belong to the same class as the centroid, whereas negative objects belong to a different class:

$$\Pi_i(r) = \frac{\Gamma_i(r)}{r}.$$

- *Optimal depth* Ω. The optimal depth is the maximum depth where for all depths r less than or equal to Ω the purity is greater than or equal to a given threshold $p_{\min}$. The optimal depth defines the maximum size of a potential cluster with a certain minimum purity. Note that it is straightforward to derive the corresponding radius from a given depth, that is, $d(x_i, x_{l_r})$:

$$\Omega_i(p_{\min}) = \max\{r | 1 \leq r' \leq r \wedge \Pi_i(r') \geq p_{\min}\}.$$

Furthermore, we introduce a final parameter Ψ for the overall coverage, which is part of the termination criterion for the algorithm. It represents the sum of all coverages of the chosen clusters. Once this threshold is reached, the algorithm stops.

6.2.4 Cluster Candidates and the Clustering Algorithm

Using the above properties we can already clarify the (automatic) clustering procedure. Starting from a user-defined value for parameter purity $\Pi = p_{\min}$ and the stopping criterion Ψ, we can compute values for parameters optimal depth Ω and coverage Γ for each potential cluster. The best cluster is identified as the one with the highest coverage. This cluster then "covers" all objects that are within its radius. These objects are then discarded from the data-set and the cluster-selection process can start again in a sequential covering manner, based on the reduced set of remaining objects. The termination criterion of the algorithm is based on the accumulated coverage of identified clusters: once it exceeds a certain threshold given by the user-defined overall coverage Ψ, the algorithm stops. Thus, the numbers of clusters is implicitly determined by the algorithm as new clusters are being added as long as the coverage is below Ψ. The basic algorithm is outlined in Table 6.1.

Although the clustering scheme as listed in Table 6.1 does not incorporate user interaction, it is fairly easy to integrate: the algorithm determines the (numerically) best Neighborgram (line (6)) and adds it to the set of clusters. However, instead of simply adding the Neighborgram, the ranking (according to $\Gamma_i(\Omega_i)$) can be used to suggest discriminative Neighborgrams to the user, who might be interested in picking another (second choice) Neighborgram or changing the cluster boundaries. Please note that the identification of good Neighborgrams is always bound to an appropriate visualization of the underlying objects in the cluster. For example, as we will also see later in Section 6.3, if the objects represent molecular drug candidates, an accompanying visualization of the molecular structures of the objects in one cluster can help the user to judge if these objects do indeed have something in common or if they are just artifacts in the data.

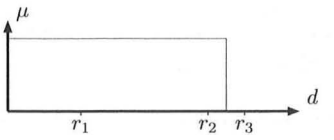
The basic clustering algorithm in Table 6.1 removes objects once they are covered by a cluster. This effect might be desirable for objects lying close to the center of the new cluster but it will reduce accuracy in areas further away from the cluster center. We therefore introduce the notion of *partial coverage* using fuzzy membership functions, which allows us to model a degree of membership of a particular object to a cluster. The next section will present the membership functions used.

Table 6.1 The basic Neighborgram clustering algorithm (Berthold, Wiswedel and Patterson, 2005).

(1)	$\forall x_i$: $c(x_i)$ is class of interest $\Rightarrow$ compute NG_i
(2)	$\forall \text{NG}_i$: compute $\Omega_i(p_{\min})$
(3)	$\forall \text{NG}_i$: compute $\Gamma_i(\Omega_i)$
(4)	$s := 0$
(5)	while $s < \Psi$
(6)	$i_{\text{best}} = \arg\max_i\{\Gamma_i(\Omega_i)\}$
(7)	add $\text{NG}_{i_{\text{best}}}$ to list of clusters, add $\Gamma_{i_{\text{best}}}(\Omega_{i_{\text{best}}})$ to s
(8)	determine list of covered objects
(9)	remove them from all Neighborgrams NG_i
(10)	$\forall \text{NG}_i$: recompute $\Gamma_i(\Omega_i)$
(11)	end while

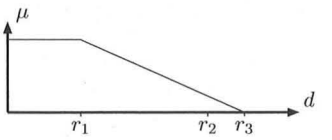
6.2.5 Membership Functions

The idea underlying the partial coverage is that each cluster is modeled by a fuzzy membership function. This function has its maximum at the centroid and declines toward the cluster boundaries. The coverage is then determined using the corresponding degrees of membership. Objects are removed to a higher degree towards the inner areas of a cluster and to a lesser degree toward the outer bounds. Figures 6.3 to 6.6 show the four membership functions we used. Note that the rectangular membership function corresponds to



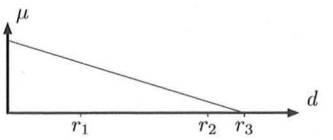
$$\mu(d) = \begin{cases} 1, & \text{if } 0 \leq d \leq \frac{r_2+r_3}{2} \\ 0, & \text{otherwise.} \end{cases}$$

Figure 6.3 The *rectangular* membership function.



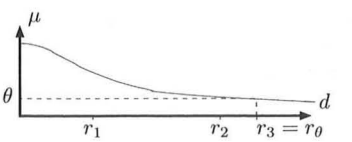
$$\mu(d) = \begin{cases} 1, & \text{if } 0 \leq d \leq r_1 \\ \frac{r_3-d}{r_3-r_1}, & \text{if } r_1 < d \leq r_3 \\ 0, & \text{otherwise.} \end{cases}$$

Figure 6.4 The *trapezoidal* membership function.



$$\mu(d) = \begin{cases} \frac{r_3-d}{r_3}, & \text{if } 0 \leq d \leq r_3 \\ 0, & \text{otherwise.} \end{cases}$$

Figure 6.5 The *triangular* membership function.



$$\mu(d) = \exp\left(-\frac{d^2}{\sigma^2}\right) \quad \text{with } \sigma^2 = -\frac{r_3^2}{\ln(\theta)}$$

Figure 6.6 The *gaussian* membership function.

the basic algorithm discussed above: objects are covered with degrees of 0 or 1 only, and are therefore removed completely when covered.

In order to describe a cluster by means of a membership function we first need to introduce three radii, which will help to specify different regions of the neighborhood:

- r_1 represents the radius of the last object with $\Pi = 1$ (last known perfect): $r_1 = \max\{r \mid \Pi_i(r) = 1\}$.
- r_2 is the last object with $\Pi \geq p_{\min}$ (last known good), that is, $r_2 = \max\{r \mid 1 \leq r' \leq r \wedge \Pi_i(r') \geq p_{\min}\}$.
- r_3 describes the first object for which $\Pi < p_{\min}$ (first known bad), that is, $r_3 = \max\{r \mid 1 \leq r' \leq r - 1 \wedge \Pi_i(r') \geq p_{\min}\}$.

These radii are sufficient to describe as shown in Figure 6.3 to 6.6 commonly used membership functions.

While the shape of the rectangular, trapezoidal and triangular membership functions are determined by the three radii, the Gaussian membership function is specified using the additional parameter θ. The inverse value of θ, r_θ, determines radius r_3. For a minimum required purity $p_{\min}$ equal to 1, the parameter θ determines the maximum degree of membership of an incorrect class for other objects in the training data (see Berthold and Diamond (1998) for details).

Using these fuzzy membership functions the clustering algorithm changes slightly. First, a degree of exposure (measuring how much is still uncovered), $\eta \in [0, 1]$, needs to be assigned to each object of the classes of interest. At the beginning this value is initialized to 1.0 for each object, that is, each object still needs to be covered completely. Subsequently this value will be decreased during clustering. A new cluster which (partly) covers an object will reduce this value accordingly. Obviously an object can only be covered until $\eta = 0$. Let $\eta(x)$ be an object's degree of exposure and $\mu_{Cluster}(d(x_i, x))$ the degree of membership to the cluster. Then the partial coverage Φ of a cluster is defined as:

$$\Phi_i(\Omega_i) = \sum_{\substack{x_{l_{r'}} \in \mathrm{NG}_i \mid 1 \leq r' \leq \Omega_i \\ \wedge c(x_{l_{r'}}) = c(x_i)}} \min\{\eta(x_{l_{r'}}),\ \mu_{Cluster}(d(x_i, x_{l_{r'}}))\}.$$

The new fuzzy version of the algorithm is shown in Table 6.2. A list of objects for the class of interest needs to be created in conjunction with their degrees of coverage (step (2)). Steps (8) and (9) of the basic algorithm are modified to incorporate the notion of partial coverage. Note that we do not need to remove covered objects from other Neighborgrams anymore, since the added degree of exposure does this implicitly.

The introduction of this concept of partial coverage improves the accuracy substantially. Experiments on publicly available data-sets from the StatLog project (Michie, Spiegelhalter, and Taylor, 1994) demonstrate that the performance of such an automatic classifier is comparable to state-of-the-art

Table 6.2 The fuzzy Neighborgram clustering algorithm (Berthold, Wiswedel, and Patterson, 2005).

(1)	$\forall x_i : c(x_i)$ is class of interest $\Rightarrow$ compute NG_i
(2)	$\forall x_i : c(x_i)$ is class of interest $\Rightarrow$ store $\eta(x_i) = 1$
(3)	$\forall \mathrm{NG}_i$: compute Ω_i
(4)	$\forall \mathrm{NG}_i$: compute $\Phi_i(\Omega_i)$
(5)	$s := 0$
(6)	while $s < \Psi$
(7)	$i_{\text{best}} = \arg\max_i\{\Phi_i(\Omega_i)\}$
(8)	add $\mathrm{NG}_{i_{\text{best}}}$ to list of clusters, add $\Phi_{i_{\text{best}}}(\Omega_{i_{\text{best}}})$ to s
(9)	recompute η for each object and
(10)	$\forall \mathrm{NG}_i$: recompute $\Phi_i(\Omega_i)$
(11)	end while

techniques (among others c4.5, k nearest neighbor, and a multi-layer perceptron). We do not discuss these experiments here but rather refer to (Berthold, Wiswedel, and Patterson, 2005). The use of fuzzy membership functions as cluster description increased the generalization ability of the classifier significantly. The best performance was always achieved using the Gaussian membership function, which, however, has one important drawback: it always produces an output since the membership value is always greater than 0. In most cases a classifier that also produces a "do not know" answer is preferable, as it allows an obviously uncertain classification to be deferred to an expert or to another system. In the following we will therefore concentrate on the other membership functions instead, also because they allow for a more intuitive visualization.

6.3 INTERACTIVE EXPLORATION

As already outlined in the introduction, the main focus of this chapter lies on the interactive exploration of such supervised fuzzy clusters. The quality measures as introduced in the previous sections allow Neighborgrams to be ranked based on their coverage given a user-defined threshold value for the purity $p_{\min}$. The system uses these values to suggest potentially interesting Neighborgrams to the user who is then able to evaluate how interesting they are. This section demonstrates the usefulness of this approach by means of some examples. We will first briefly explain the visualization technique of a Neighborgram and its cluster candidate before we show its applicability in practice on a real-world data-set from the National Cancer Institute.

6.3.1 Neighborgram Visualization

Since a Neighborgram is a one-dimensional representation of the neighborhood of an object, it is straightforward to visualize. Figure 6.7 shows the Neighborgrams for two objects of the Iris data-set (Fisher, 1936). In comparison to Figure 6.2 the figure also contains a visualization of the (trapezoidal) fuzzy membership function as proposed by the system. While the vertical axis does not have a meaning for the visualization of points (as noted earlier, we use it only to avoid overlaps), it is used to display the membership function.

Both clusters in Figure 6.7 are built for the Iris-Virginica class (points shown in black); however, the top Neighborgram suggests better clustering behavior as it covers almost all of the objects of Virginica class (the same class as the centroid), whereas the bottom Neighborgram also has objects of Iris-Versicolor class (white points) in its close neighborhood. Note also how objects of Iris-Setosa class (gray) form a nice separate cluster far away in both Neighborgrams, a fact well-known from the literature. In this case automatic ranking is likely to be a good choice; however, in a less obvious case, the user could

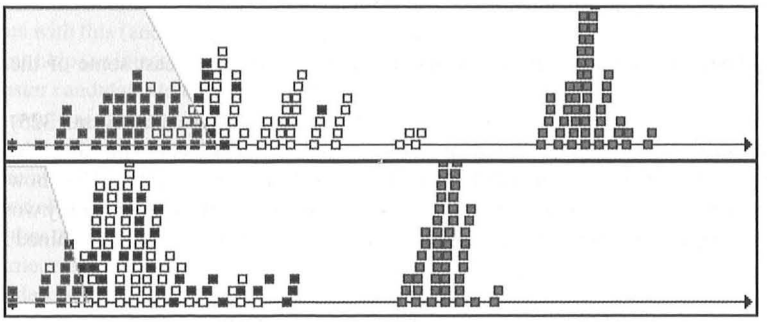


Figure 6.7 Two Neighborgrams built for the Iris data.

overrule the algorithm's choice, select individual clusters, and also modify their membership functions if so desired.

6.3.2 NCI's HIV Data

To show the usefulness of the proposed visual clustering algorithm in a real-world scenario, let us look at the application of Neighborgrams on a well-known data-set from the National Cancer Institute, the DTP AIDS Antiviral Screen data-set. The screen utilized a soluble formazan assay to measure protection of human CEM cells from HIV-1 infection. All compounds in the data-set were tested on their protection of the CEM cell; those that did not provide at least 50 % protection were labeled as confirmed inactive (**CI**). All others were tested in a second screening. Compounds that provided protection in this screening, too, were labeled as confirmed active (**CA**), the remaining ones as moderately active (**CM**). Available online (National Cancer Institute, 2005) are screening results and chemical structural data on compounds that are not protected by a confidentiality agreement. Available are 41 316 compounds of which we have used 36 045[2]. A total of 325 belongs to class **CA**, 877 are of class **CM** and the remaining 34 843 are of class **CI**. Note the class distribution for this data-set is very unbalanced, there are about 100 times as many inactive compounds (**CI**) as there are active ones (**CA**). The focus of analysis is on identifying internal structures in the set of active compounds as they showed protection to the CEM cells from an HIV-1 infection.

This data-set is a very typical application example of the Neighborgram classifiers: although it is a relatively large data-set, it has an unbalanced class distribution with the main focus on a minority class.

In order to generate Neighborgrams for this data-set, a distance measure needs to be defined. We initially computed Fingerprint descriptors (Clark, 2001), which represent each compound through a 990-dimensional bit string. Each bit represents a (hashed) specific chemical substructure of interest. The used distance metric was a Tanimoto distance, which computes the number of bits that are different between two vectors normalized over the number of bits that are turned on in the union of the two vectors. The Tanimoto distance is often used in cases like this, where the used bit vectors are only sparsely occupied with 1s.

NCI lists a small number (75) of known active compounds, which are grouped into seven chemical classes:

- azido pyrimidines;
- pyrimidine nucleosides;
- heavy metal compounds;
- natural products or antibiotics;
- dyes and polyanions;
- purine nucleosides;
- benzodiazepines, thiazolobenzimidazoles, and related compounds.

One would expect that a decent clustering method would retrieve at least some of these classes of compounds.

Therefore, we constructed Neighborgrams for all of the active compounds (overall 325) and used the system to rank these Neighborgrams based on their coverage. Figure 6.8 shows the biggest cluster that the algorithm encountered using a purity of 90 % and building clusters for class **CA**. Note how the next two rows show Neighborgrams for compounds of the same cluster, both of them with slightly worse computed *purity* and *coverage*. At first we were surprised to see that none of the compounds contained in this cluster

[2]For the missing compounds we were unable to generate the used descriptors.

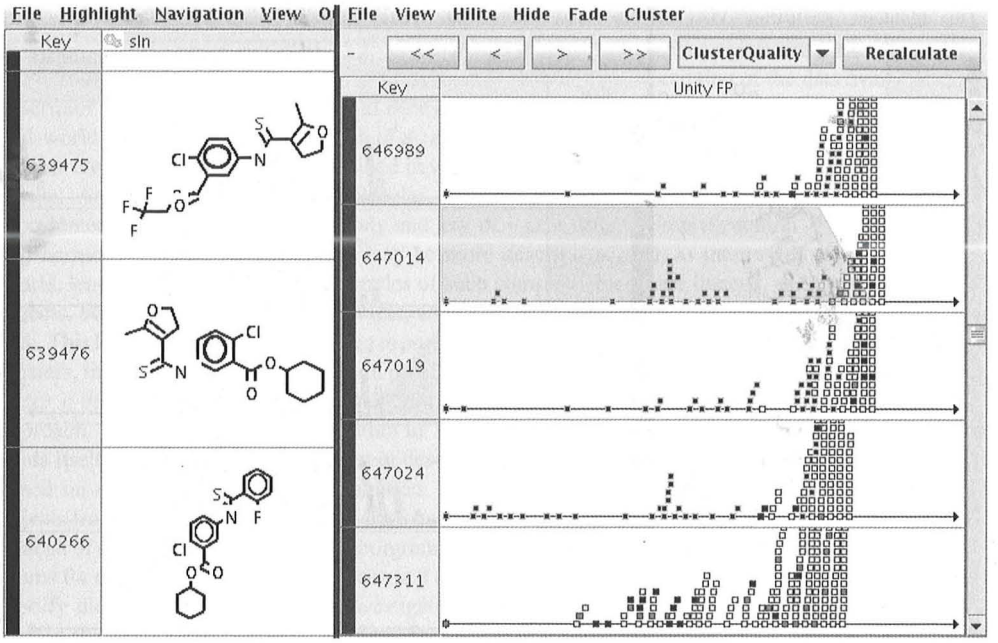


Figure 6.8 The first cluster of the NIH-Aids data centered around compound #647014. On the right the Neighborgrams in unity-fingerprint space (black=**CA**, gray=**CM**, white=**CI**), on the left a view showing some of the structures contained in this cluster.

fall in any of the classes of active compounds listed on NIH's Web site (National Cancer Institute, 2005). As it turns out when looking at the corresponding structures, this cluster covers m-acylaminobenzamides which probably all inhibit folic acid synthesis, but are likely to be too toxic and hence not very interesting as active compounds to fight HIV. This is therefore a nice example of a cluster that a chemist might discard as 'useful but not very interesting for the current task at hand." The clustering algorithm has no insights other than numerical cluster measures and therefore would assign a high ranking value to this cluster without any expert interaction.

Subsequent clusters reveal groupings very much in line with the classes listed above, one particular example is shown in Figure 6.9. Here the group of "dyes and polyanions" are grouped together in a nice cluster with almost perfect purity (two inactive compounds are covered as well). Figure 6.10 shows another example, this time grouping together parts of the group of "azido pyrimidines," probably one of the best-known classes of active compounds for HIV.

Experiments with this (and other similar) data-sets showed nicely how the interactive clustering using Neighborgrams helps to inject domain knowledge in the clustering process and how Neighborgrams help promising cluster candidates to be quickly inspected visually. Without the additional display of chemical structure this would not have worked as convincingly. It is important to display the discovered knowledge in a "language" the expert understands.

In our experiments we were confronted with the problem of which descriptor to use. In the previous experiments we always used unity fingerprint descriptors in conjunction with the Tanimoto distance. We have ignored that there may be more than just this single description available. However, particularly in the field of molecular data analysis there are numerous ways to describe molecules and it is hardly ever known which descriptor is best. This problem relates to the learning in parallel universes, which will be addressed in the next section.

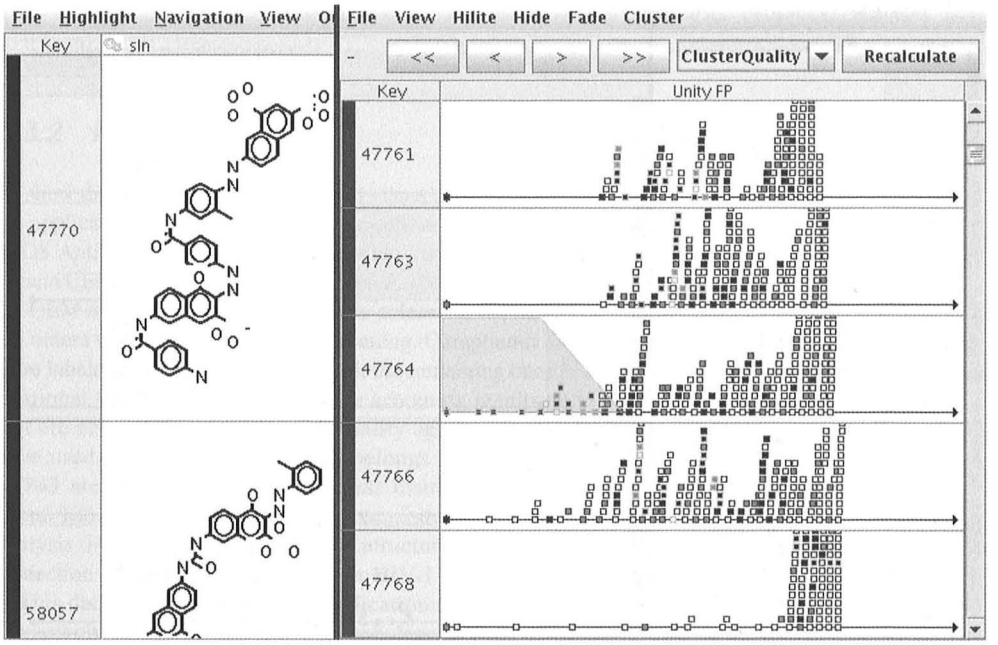


Figure 6.9 Another cluster of the NIH-Aids data centered around compound #47764 (right: Neighborgrams (black=**CA**, gray=**CM**, white=**CI**), left: structures). This cluster nicely covers one of the classes of active compounds: dyes and polyanions.

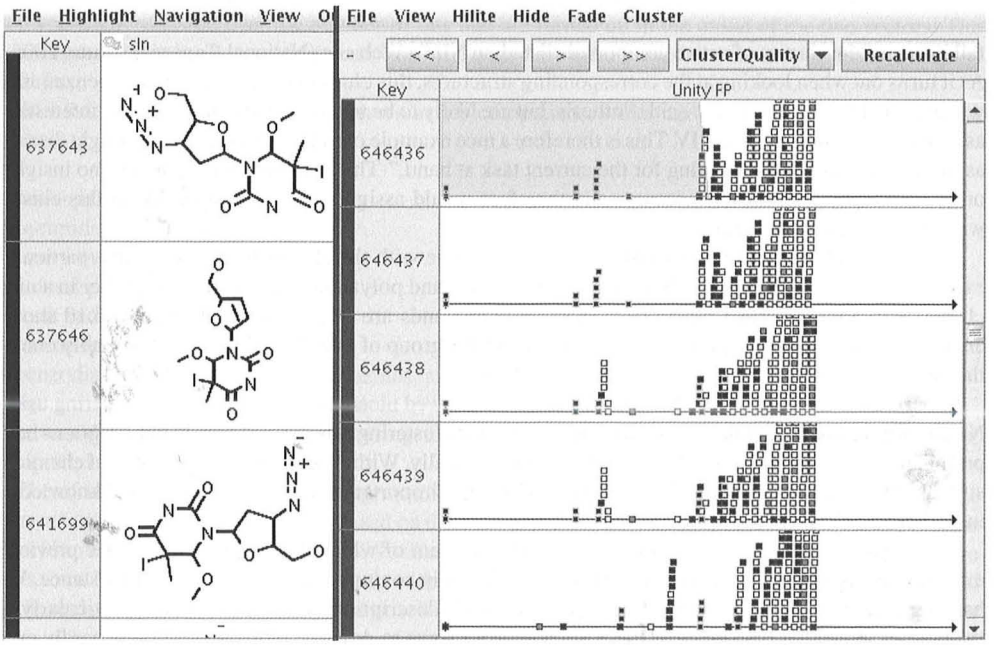


Figure 6.10 Another cluster of the NIH-Aids data centered around compound #646436 (right: Neighborgrams (black=**CA**, gray=**CM**, white=**CI**), left: structures). This cluster nicely covers part of one of the most well-known classes of active compounds: azido pyrimidines.

6.4 PARALLEL UNIVERSES

In the foregoing experiment we assumed that there is an adequate description of the data available. This descriptor was expected to comprise all necessary information to classify an object. However, in many real-world applications, the generation of an appropriate descriptor is far from trivial as the underlying objects are complex and can be described in various ways, focusing on different aspects of the object's nature. An example, other than molecules, are musical songs, i.e., audio streams, which can be represented based on dynamics, melody and key or – as a different representation – based on rhythm and harmony. A third representation may be more descriptive, such as interpreter, position in music charts, length, and so on. Further examples of such complex objects are images, and three-dimensional objects. For the learning it is often unclear, which of the available descriptors are optimal for any given task. This leads to the notion of *learning in parallel universes*, where we can find interesting patterns, e.g., clusters, in different descriptor spaces in parallel. Wiswedel and Berthold (2006), for instance, applied a fuzzy c-means algorithm to data described in parallel universes. However, this is an unsupervised approach. When looking at the algorithm in Table 6.1, one notes that the Neighborgram methodology lends itself naturally to handling different descriptor spaces: the clusters do not interact with each other based on any universe-specific information. Besides the fact that a chosen cluster removes covered objects from consideration there is no obvious need for two clusters to originate from the same universe. Instead of constructing just one Neighborgram for each object of interest, we can easily create Neighborgrams for each available descriptor space and consider these as potential cluster candidates. We can then modify the clustering algorithm to investigate all Neighborgrams in all feature spaces in parallel and choose the best cluster among all universes. Covered objects will subsequently be removed from all universes and the result is a set of clusters, spread out over different feature spaces.

Figure 6.11 shows an example for the HIV data from the previous section. However, rather than just having one fingerprint descriptor as before, we computed two other descriptors of the underlying compounds. Firstly, we generated an AtomPair fingerprint descriptor, a 1200-dimensional bit vector, which encodes the presence or absence of certain pairs of atoms in the molecule. The second one is a VolSurf descriptor (Cruciani, Crivori, Carrupt and Testa, 2000), i.e., a 56-dimensional numerical feature vector encoding molecular properties such as molecular weight and two-dimensional numerical descriptions describing properties of the three-dimensional structure. Distances were calculated using the Euclidean distance in VolSurf space and Tanimoto distance for both fingerprints (Unity and AtomPair). The left column in the figure shows Neighborgrams in AtomPair space, the middle column Neighborgrams in Unity space, whereas the Neighborgrams in the right column are constructed using the VolSurf descriptor.

Note how the Neighborgrams in the first row differ substantially although they represent the neighborhood of the same object (#662427). The Unity fingerprint descriptor (middle) suggests for this compound

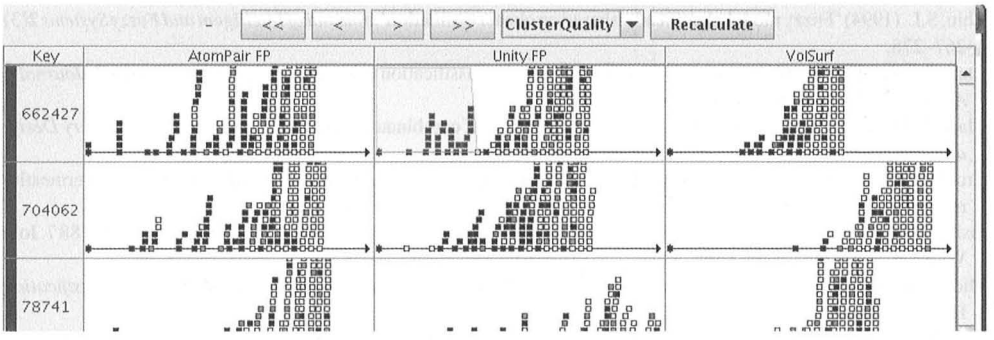


Figure 6.11 Neighborgrams for the NIH-Aids data in parallel universes. The left column in a AtomPair fingerprint space (1200-dimensional bit vectors), the middle column Unity fingerprint space (990-dimensional bit vectors), and the right column 56-dimensional Euclidean space, the VolSurf descriptor. Note how the Neighborgrams in the top row differ, although they represent the neighborhood of the same compound, however in different universes.

the best cluster as it covers about 25 active molecules (black points). However, the neighborhood of this compound in the VolSurf space (right) contains only few actives (less than 10) and in the AtomPair fingerprint space about 15. This example demonstrates that the definition of a cluster depends on the underlying object description. The user can consequently inspect the cluster and potentially gain new insights as to why objects group in one universe but not in another. Especially for data-sets that involve structural descriptions of molecules it is hardly ever known which descriptor is optimal for a particular problem. The final outcome of the clustering algorithm itself is a set of clusters originating from different feature spaces.

6.5 DISCUSSION

In this chapter we discussed a supervised approach to identify and visually explore a set of fuzzy clusters. We used a one-dimensional data structure, a so-called Neighborgram, to depict local neighborhoods of each object. Constructing Neighborgrams for all objects of interest, for example, all objects of a particular class, and deriving potential cluster candidates from them, allowed us to rank these Neighborgrams. An automatic clustering algorithm sequentially accepts the top-ranked cluster and removes all objects covered by this cluster from consideration. More important, however, is that the accompanying visualization of a Neighborgram provides a powerful way to explore the proposed cluster selection and enables the user to inject domain knowledge into the clustering process by accepting, discarding, or fine-tuning potential cluster candidates. Using a real-world data-set from a bioinformatics application we demonstrated how this method of visual exploration supports the user in finding potentially interesting groupings in the data. The described technique provides a tool for interactive exploration of large data-sets, allowing for truly intelligent data analysis.

REFERENCES

Ankerst, M., Elsen, C., Ester, M. and Kriegel, H.P. (1999) 'Visual classification: An interactive approach to decision tree construction'. *Proceedings of the Fifth, A.C.M. SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 392–396.

Berthold, M.R. (2003) 'Mixed fuzzy rule formation'. *International Journal of Approximate Reasoning (IJAR)* **32**, 67–84.

Berthold, M.R. and Diamond, J. (1998) 'Constructive training of probabilistic neural networks'. *Neurocomputing* **19**, 167–183.

Berthold, M.R., Wiswedel, B. and Patterson, D.E. (2005) 'Interactive exploration of fuzzy clusters using neighborgrams'. *Fuzzy Sets and Systems* **149**(1), 21–37.

Bezdek, J.C. (1981) *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York.

Chiu, S.L. (1994) 'Fuzzy model identification based on cluster estimation'. *Journal of Intelligent and Fuzzy Systems* **2**(3), 267–278.

Chiu, S.L. (1997) 'An efficient method for extracting fuzzy classification rules from high dimensional data'. *Journal of Advanced Computational Intelligence* **1**(1), 31–36.

Clark, R.D. (2001) 'Relative and absolute diversity analysis of combinatorial libraries' *Combinatorial Library Design and Evaluation* Marcel Dekker New York pp. 337–362.

Cruciani, G., Crivori, P., Carrupt, P.A. and Testa, B. (2000) 'Molecular fields in quantitative structure-permeation relationships: the VolSurf approach'. *Journal of Molecular Structure* **503**, 17–30.

Fisher, R.A. (1936) 'The use of multiple measurements in taxonomic problems' *Annual Eugenics, II*, pp. 179–188 7. John Wiley & Sons, Inc., New York.

Michie, D., Spiegelhalter, D.J. and Taylor, C.C. (eds) (1994) *Machine Learning, Neural and Statistical Classification*. Ellis Horwood Limited Chichester, UK.

National Cancer Institute (2005) http://dtp.nci.nih.gov/docs/aids/aids_data.html.

Wiswedel, B. and Berthold, M.R. (2007) 'Fuzzy clustering in parallel universes'. *International Journal of Approximate Reasoning* (in press).

Yager, R.R. and Filev, D.P. (1994) 'Approximate clustering via the mountain method'. *IEEE Transaction on Systems, Man, and Cybernetics* **24**(8), 1279–1284.