First publ. in: Fuzzy Sets and Systems 149 (2005), pp. 21-37

Interactive exploration of fuzzy clusters using Neighborgrams

Michael R. Berthold[a, b, *], Bernd Wiswedel[a, b], David E. Patterson[b]

[a] *Department of Computer and Information Science, University of Konstanz, Box M712, 78457 Konstanz, Germany*
[b] *Data Analysis Research Lab, Tripos Inc., USA*

Abstract

We describe an interactive method to generate a set of fuzzy clusters for classes of interest of a given, labeled data set. The presented method is therefore best suited for applications where the focus of analysis lies on a model for the minority class or for small to medium-sized data sets. The clustering algorithm creates one–dimensional models of the neighborhood for a set of patterns by constructing cluster candidates for each pattern of interest and then chooses the best subset of clusters that form a global model of the data. The accompanying visualization of these neighborhoods allows the user to interact with the clustering process by selecting, discarding, or fine–tuning potential cluster candidates. Clusters can be crisp or fuzzy and the latter leads to a substantial improvement of the classification accuracy. We demonstrate the performance of the underlying algorithm on several data sets from the StatLog project and show its usefulness for visual cluster exploration on the Iris data and a large molecular dataset from the National Cancer Institute.

Keywords: Cluster analysis; Visual exploration; Neighborgram

1. Introduction

The analysis of large data sets has gained considerable interest over the past decade or so. Modern technology enables recording vast amounts of data that need to be analyzed in order to reveal interesting relationships. Methods from diverse disciplines have been combined to help analyze such data sets. An introduction to the most commonly used methods can be found in [1]. In recent years, visualization

* Corresponding author. Department of Computer and Information Science, University of Konstanz, Box M712, 78457 Konstanz, Germany.
E-mail address: michael.berthold@uni-konstanz.de (M.R. Berthold).

techniques that summarize data have also emerged allowing for interactive exploration of data or models, [6] summarizes some of the techniques.

Many different approaches to build interpretable models for classification have been proposed in the past. Although the traditional cluster algorithm works on unsupervised data sets, extensions also allow for the building of cluster models to 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. However, even without explicit translation into fuzzy rules, fuzzy clusters are well suited for presentation of the resulting classification model to the user. In [4] an extensive overview of fuzzy cluster algorithms and also their use for classification tasks is presented.

Most clustering methods iterate over the available data in order to minimize a particular objective function. Many of these algorithms attempt to find a cluster center by continuously adjusting the location of a cluster representative; many of the well-known fuzzy clustering algorithms are of this type. Adjusting the cluster centers in an iterative manner (often performed by gradient descent procedures) makes sense if we deal with huge amounts of data. The computational effort is reasonable—at least in comparison to, for example, the training of a multi-layer perceptron. However, the final solution is often suboptimal and in addition, the number of clusters must be known in advance. It is equally necessary that the mean of a subset of patterns is computed in order to represent the center vector of each cluster. Occasionally this averaging procedure also makes subsequent interpretation of the cluster centers difficult.

Approaches which find the cluster centers directly have also been proposed. One example is the DDA algorithm, a constructive training algorithm for probabilistic neural networks [2]. The algorithm picks the cluster centers directly from the training patterns. A new cluster is introduced whenever the current network cannot model the newly encountered pattern. The new cluster is positioned at this pattern. This approach does not find optimal cluster centers, since the positions of the cluster centers depend on the order in which the training patterns are processed but it is straightforward to use and very fast. Additionally, since the cluster centers are picked from the training examples, they remain interpretable. Another algorithm from this category is the mountain method [11], which picks cluster centers from the predifined data sequentially, at each step optimizing a fitness function, the so-called mountain function. This method is based on an underlying grid and a non-intuitive neighborhood parameter α, which the user has to specify a priori.

The algorithm described here belongs to the latter category of algorithms mentioned above. But rather than using a heuristic or greedy algorithm to select example patterns as cluster representatives, the proposed method analyzes the neighborhood of each cluster candidate and picks (at each step) the optimal cluster representative directly (similar to the mountain method [11]). Such a complete and hence computationally expensive approach obviously only works for all classes of a medium-sized 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 patterns 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.

This paper deals with a representation of such data by computing a so-called *Neighborgram* for each example of the class(es) of interest. A Neighborgram is a summarization of the neighborhood of a pattern, which allows an interpretable view on the underlying data. The proposed algorithm then finds clusters in a set of such Neighborgrams based on an optimality criteria. Since Neighborgrams are easy to interpret, the

algorithm can also be used to visually suggest clusters to the user, who might be interested in influencing 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 patterns 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 patterns do require the availability of an underlying feature representation in order to continuously compute and update the cluster centers.

This paper 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 3 discusses experiments on some publicly available datasets from the StatLog project and demonstrates that the fully automated clustering algorithm achieves results comparable to those of state-of-the-art techniques. We continue by sketching the usefulness of the visual clustering procedure and conclude with a discussion of possible extensions and future directions of research.

2. Neighborgram clustering

In this section we first introduce the concept of Neighborgrams, the underlying data structure of the proposed clustering algorithm. Afterwards we describe the clustering algorithm itself along with a discussion of different membership functions.

We assume a set of training patterns 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, $c(x_i) = k$, $1 \leqslant k \leqslant C$.

2.1. Neighborgrams

A Neighborgram is a one–dimensional model of the neighborhood of a chosen pattern, which we will call the *centroid*. Other patterns are mapped into the Neighborgram depending on their distance to this centroid. Essentially, a Neighborgram summarizes the neighborhood of the centroid through a ray onto which the closest neighbors of the centroid are plotted. Obviously, mapping all patterns onto the ray would be complicated and the visualization would lose its clarity. Therefore, we introduce a parameter R which determines the maximum number of patterns 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:

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

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

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

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

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

[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 patterns.

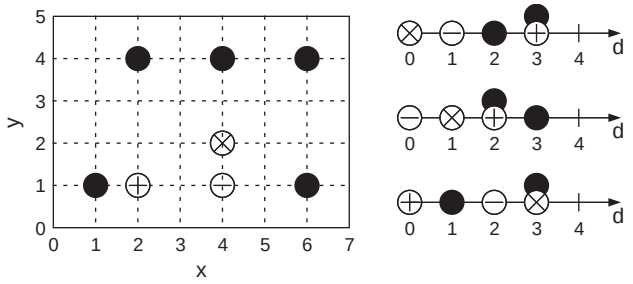


Fig. 1. An example feature space with three Neighborgrams.

Note that $l_1 = i$, because $d(x_i, x_i) = 0$ for all i, that is, each pattern is closest to itself. Note also that this list is not necessarily a unique representation since it is possible for two entries in the list to 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 case of large data sets the computation of Neighborgrams for each training pattern 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 patterns 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 case of large data sets. This complexity estimate is derived as follows: For each pattern ($O(M)$) and for each Neighborgram ($O(M')$) do an Insertion Sort into a list of R patterns ($O(R)$). If the size R of the Neighborgrams is closer to the overall number of patterns 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.

Example 2.1. Fig. 1 shows an example with three patterns of the positive class (empty circles) and four negative patterns (solid circles). The Manhattan distance is used to determine distances between patterns, which can be calculated by counting vertical or horizontal steps that have to be taken to move along the grid. Between $\otimes$ and $\oplus$ there are two steps horizontally and one vertically. Therefore the distance is 3. The three Neighborgrams on the right show the neighborhoods of the positive patterns for $R = 5$. The lists defining the Neighborgrams can also be written as:

$\mathrm{NG}_{\otimes} = [\otimes, \ominus, \bullet, \oplus, \bullet]$
$\mathrm{NG}_{\ominus} = [\ominus, \otimes, \oplus, \bullet, \bullet]$
$\mathrm{NG}_{\oplus} = [\oplus, \bullet, \ominus, \otimes, \bullet]$

We will continue to use this example in the following section to illustrate the basic algorithm.

2.2. The basic clustering algorithm

The key idea underlying the clustering algorithm is that each pattern 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 find the "best" cluster at each step. The result is a subset of all possible clusters which covers a sufficient number of patterns.

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 patterns covered by this cluster, and
(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, and what removing covered patterns 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.

2.3. Neighborgram properties

In Section 2.1 an ordered list was suggested as representation for a Neighborgram. This list contains patterns which are ordered according to their distance to the centroid. The length of the list is determined by the parameter R:

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

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

- *Coverage* Γ: The default coverage of a cluster with a certain depth $r \leqslant R$ determines how many positive patterns it "explains", that is, the number of positive patterns that fall within its radius:

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

  Example 2.2. For pattern $\oplus$ in Fig. 1 the coverage is: $\Gamma_\oplus(1) = 1$, $\Gamma_\oplus(2) = 1$, $\Gamma_\oplus(3) = 2$, and $\Gamma_\oplus(5) = 3$.

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

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

  Example 2.3. For Fig. 1 the purity for the pattern $\oplus$ depending on depth r would be: $\Pi_\oplus(1) = 1$, $\Pi_\oplus(2) = \frac{1}{2}$, $\Pi_\oplus(3) = \frac{2}{3}$, and $\Pi_\oplus(5) = \frac{3}{5}$.

- *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

Table 1
The basic Neighborgram clustering algorithm

(1)	$\forall x_i$: $c\,(x_i)$ is class of interest $\Rightarrow$ compute NG_i
(2)	$\forall \mathrm{NG}_i$: compute $\Omega_i\,(p_{\min})$
(3)	$\forall \mathrm{NG}_i$: 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 $\mathrm{NG}_{i_{\text{best}}}$ to list of clusters, add $\Gamma_{i_{\text{best}}}\left(\Omega_{i_{\text{best}}}\right)$ to s
(8)	determine list of covered patterns
(9)	remove them from all Neighborgrams NG_i
(10)	$\forall \mathrm{NG}_i$: recompute $\Gamma_i\,(\Omega_i)$
(11)	end while

corresponding radius from a given depth, that is $d(x_i, x_{l_r})$.

$$\Omega_i(p_{\min}) = \max\left\{r \mid 1 \leqslant r' \leqslant r \wedge \Pi_i(r') \geqslant p_{\min}\right\}.$$

Example 2.4. In Fig. 1 we find for pattern $\ominus$ and $p_{\min} = 1.0$ that $\Omega_{\ominus}(1.0) = 2$. For $p_{\min} = \frac{2}{3}$ we get $\Omega_{\ominus}\left(\frac{2}{3}\right) = 3$.

- *Minimum Size* Λ: The minimum size a cluster must have. Computing purity as described above has the disadvantage that for noisy data sets many clusters will not extend as far as they could because an early encounter of a pattern of the wrong class will set the optimal depth Ω very close to the centroid. To avoid this, we introduce a parameter Λ, which allows us to specify a minimum number of patterns in a neighborhood before the purity Π and the optimal depth Ω are determined. Early experiments with noisy data sets have shown a decrease in number of clusters and better generalization ability.

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.

2.4. Cluster candidates and the clustering algorithm

We can now specify more clearely what we mean by ranking clusters and removing covered patterns. Starting from a user-defined value for parameter purity $\Pi = p_{\min}$ we can compute values for parameters optimal depth Ω and coverage Γ for each cluster. The best cluster is the one with the highest coverage. This cluster then "covers" all patterns that are within its radius. These patterns are then discarded from the data set and the cluster-selection process can start again, based on the reduced set of patterns to be explained. Table 1 summarizes the basic algorithm.

It is obvious that the basic algorithm sketched here is very strict—a pattern will be completely removed from any further consideration as soon as it falls within the optimal radius for just one single cluster. This effect might be desirable for patterns 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 pattern to a cluster. The next section will present the used membership functions.

2.5. Membership functions

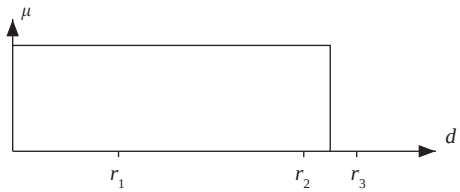
The idea underlying the partial coverage is that each cluster is modeled by a fuzzy membership function. This function has its maximum at the centroid and declines towards the cluster boundaries. The coverage is then determined using the corresponding degrees of membership. Patterns are removed to a higher degree towards the inner areas of a cluster and to a lesser degree towards the outer bounds. The following figures show the four membership functions we used. Note that the rectangular membership function corresponds to the basic algorithm discussed above: patterns 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 pattern with $\Pi = 1$ (last known perfect), $r_1 = \max\{r \mid \Pi_i(r) = 1\}$.
- r_2 is the last pattern with $\Pi \geqslant p_{\min}$ (last known good), that is, $r_2 = \max\left\{r \mid 1 \leqslant r' \leqslant r \wedge \Pi_i(r') \geqslant p_{\min}\right\}$.
- r_3 describes the first pattern for which $\Pi < p_{\min}$ (first known bad), i.e., $r_3 = \max\left\{r \mid 1 \leqslant r' \leqslant r - 1 \wedge \Pi_i(r') \geqslant p_{\min}\right\}$.

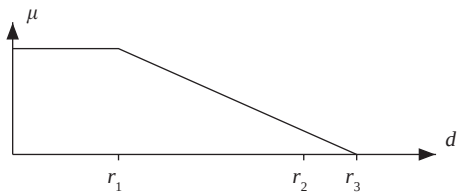
These radii are sufficient to describe the following commonly used membership functions.

The **Rectangular** Membership Function.



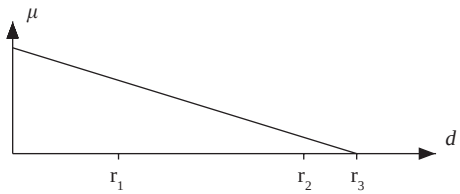
$$\mu(d) = \begin{cases} 1, & \text{if } 0 \leqslant d \leqslant \frac{r_2 + r_3}{2} \\ 0, & \text{otherwise.} \end{cases}$$

The **Trapezoidal** Membership Function.



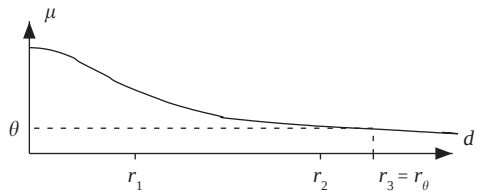
$$\mu(d) = \begin{cases} 1, & \text{if } 0 \leqslant d \leqslant r_1 \\ \frac{r_3 - d}{r_3 - r_1}, & \text{if } r_1 < d \leqslant r_3 \\ 0, & \text{otherwise.} \end{cases}$$

The **Triangular** Membership Function.



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

The **Gaussian** Membership Function.



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

While the shape of the rectangular, trapezoidal, and triangular membership function is 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 patterns in the training data (see [2] 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 pattern of the classes of interest. At the beginning this value is initialized to 1.0 for each pattern, that is, each pattern still needs to be covered completely. Subsequently this value will be decreased during clustering. A new cluster which (partly) covers a pattern will reduce this value accordingly. Obviously a pattern can only be covered until $\eta = 0$. Let $\eta(x)$ be a pattern's degree of exposure and $\mu_{\text{Cluster}}(d(x_i, x))$ the degree of membership to the cluster. Then the Partial Coverage Φ of a cluster is defined as

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

The introduction of this concept of partial coverage improved the clustering substantially—as we will discuss later in the next section. The new fuzzy version of the algorithm is shown in Table 2. A list of patterns 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 patterns from other Neighborgrams anymore, since the added degree of exposure does this implicitly.

3. Results

We used four standard data sets from the StatLog project [7] to evaluate the generalization ability of the clusters generated by the presented algorithm. In order to be able to compare the new method to published results, we generated clusters for all classes, i.e. for each class we built Neighborgrams to discriminate against all other classes. The classification outcome (the predicted class) was determined by the maximum of the sum of the degrees of membership from all Neighborgrams of all classes. Note that for all membership functions except the Gaussian one it is possible to have an output of 0.0 for all classes, in which case the default class was predicted, that is, the class with the highest apriori occurrence frequency.

Table 2
The fuzzy Neighborgram clustering algorithm

(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) $\quad i_{\mathrm{best}} = \arg\max_i \{\Phi_i(\Omega_i)\}$
(8) $\quad$ add $\mathrm{NG}_{i_{\mathrm{best}}}$ to list of clusters,
$\quad$ add $\Phi_{i_{\mathrm{best}}}\left(\Omega_{i_{\mathrm{best}}}\right)$ to s
(9) $\quad$ recompute η for each pattern and
(10) $\quad \forall \mathrm{NG}_i$: recompute $\Phi_i(\Omega_i)$
(11) end while

Table 3
Used data sets and misclassification rates

Name	Dimension	#Classes	Size (trn—tst)
SatImage	36	6	(4435, 2000)
Diabetes	8	2	(768, 12-fold)
Segment	19	7	(2310, 10-fold)
DNA	180 (binary)	3	(2000, 1186)

Name	c4.5	kNN	MLP	PNN	DDA
SatImage	15.0	9.4	13.9	9.8	8.9
Diabetes	27.0	32.4	24.8	24.9	24.1
Segment	4.0	7.7	5.4	3.5	3.9
DNA	7.6	14.6	8.8	10.5	16.4

Name	NG-T	NG-R	NG-Δ	NG-G (θ)
SatImage	14.4	15.9	9.9	8.1 ($1.8 \cdot 10^{-3}$)
Diabetes	29.8	31.4	27.1	24.4 ($3.0 \cdot 10^{-1}$)
Segment	8.1	9.7	3.9	3.5 ($1.0 \cdot 10^{-6}$)
DNA	25.7	28.0	19.1	16.6 ($1.0 \cdot 10^{-12}$)

3.1. Experiments

From the StatLog project [7] the four data sets listed on top of Table 3 were taken. We followed the procedure described there and either used the specified division in training and test data or applied *n*–fold cross–validation.

Results for the Neighborgram classifier are shown at the bottom of Table 3 along with results for the decision tree algorithm c4.5, *k*-nearest neighbor, and a multi-layer perceptron in the middle (all results from [7]). In addition we list results for probabilistic neural networks [9], trained by the original algorithm (PNN) where each neuron covers one training example and generated constructively through the DDA algorithm [2]. The table on the bottom shows the classification error for various membership functions:

Table 4
Results for different ways to determine the output

Name	NG-Δ		
	Max	Sum	wSum
SatImage	10.1	9.9	10.0
Diabetes	26.7	27.1	26.6
Segment	3.8	3.9	4.5
DNA	21.1	19.1	19.5

NG-T = trapezoidal, NG-R = rectangular, NG-Δ = triangular, NG-G = Gaussian.[2] All results are using the maximum summed degree of membership to make a class decision (sum).

Note how in three cases (SatImage, Diabetes and Segment) the classification accuracies of the triangular Neighborgram classifier (NG-Δ) and the Gaussian one (NG-G) compare nicely to the other methods (and also the remainder of the algorithms reported in the StatLog project). For the DNA data set, however, the NG classifier performs considerably worse no matter what membership function is used. This is mostly due to the restrictive, local nature of the clusters used by the NG clustering algorithm. In case of the DNA data set the generated NG clusters report no class for over 20% of the test patterns,[3] indicating an insufficient coverage of the feature space. Considering that this is a binary problem, it is not surprising that a distance based method fails to perform well, as can also be seen from the mediocre results by the other local methods, kNN and DDA.

Note that the Neighborgram classifier using Gaussian membership functions shows superior performance to almost all other algorithms, especially the neighborhood based algorithms, which is due to the better, i.e. non-greedy selection of cluster centers. In addition, the Gaussian membership function produces a non-zero output and therefore always predicts a class. However, in most cases a classifier that also produces a "do not know" answer is preferable, since they allow for deferral of an obviously uncertain classification to an expert or to another system. In the following we will concentrate on the other membership functions instead, also because they allow for a more intuitive visualization.

Additional experiments were conducted to investigate how the used classification computation affected the outcome. Table 4 lists the results on the same four data sets for the triangular membership function and three different ways to determine the predicted class.

- Standard (max): i.e. use maximum degree of membership (unweighted),
- sum (sum): sum up degrees of membership for each class and determine maximum, and
- weighted sum (wSum): sum up all weighted degrees of membership for each class and determine maximum.

As before, whenever the classifier did not generate a prediction, the default class was used.

Using the proposed method as a standard clustering algorithm clearly achieves satisfactory results and could therefore be used as an automatic classifier. However, as we have mentioned before, the main focus

[2] The parameter θ is given in parenthesis.

[3] Actually, the Gaussian membership function always produces an output, so this statement holds only for the three remaining membership functions.

of Neighborgrams is their ability to visualize cluster candidates. In the following we discuss this property in more detail.

4. Visual clustering

One focus of our research was on a very interesting extension: The fact that a Neighborgram requires only one dimension, i.e. the distance to the centroid, offers the opportunity to visualize the corresponding neighborhood. In addition we can also project the membership function for a given purity $p_{\min}$ onto the Neighborgram, which enables the user to select and fine-tune potential cluster candidates. In combination with the proposed clustering technique described above, this results in a system which suggests potential clusters to the user who can then visually evaluate the clusters' interestingness. We first demonstrate how this can be used on the well-known Iris data set before showing its applicability in practice on a real-world data set from the National Cancer Institute.

4.1. Iris data

Fig. 2 shows two Neighborgrams for patterns of the Iris data set with the corresponding cluster membership function. The clusters are both built for the same class (Iris-Virginica, points shown in dark gray). Note how patterns of class Iris-Setosa (white) form a nice separate cluster far away in both Neighborgrams, a fact well-known from the literature. Whenever two or more patterns are too close to each other, and overlap, we decided to stack them on top of each other, so patterns can be individually selected and are highlighted in other Neighborgrams—or even other views—as well. The vertical axes therefore has no geometrical meaning, it is simply used to avoid overlaps. In turn, for the displayed membership function the vertical axes does have a meaning, i.e. the degree of membership. Note how the quality of these two clusters becomes visible immediately. The cluster on the top nicely covers almost all of the patterns of class Virginica, whereas the cluster on the bottom only covers a few examples. In this case the automatic ranking is likely a good choice but in a less obvious case the user could overwrite the algorithm's choice, select individual clusters and also modify their membership functions if so desired.

4.2. NCIs HIV data

We used a well-known data set from the National Cancer Institute, the DTP AIDS Antiviral Screen data set, to test the Neighborgram clustering algorithm. The screen utilized a soluble formazan assay to measure protection of human CEM cells from HIV-1 infection. Full details were published in [10]. Compounds able to provide at least 50% protection to the CEM cells were retested. Compounds that provided at least 50% protection on retest were listed as moderately active (CM). Compounds that reproducibly provided 100% protection were listed as confirmed active (CA). Compounds not meeting these criteria were listed as confirmed inactive (CI). Available online [5] are screening results and chemical structural data on compounds that are not covered by a confidentiality agreement. Available are 41, 316 compounds of which we have used 36, 045.[4] A total of 325 belongs to class CA, 877 are of class CM and the

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

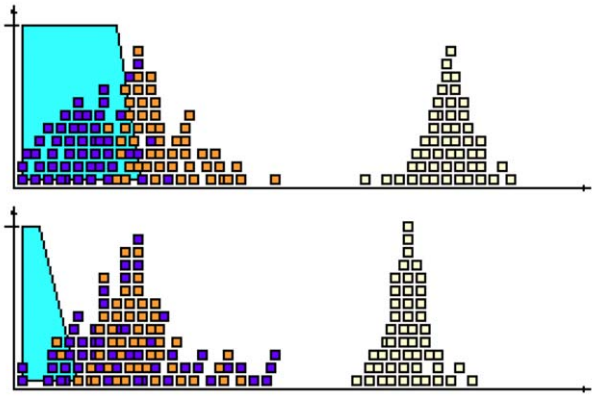


Fig. 2. Two Neighborgrams built for the Iris data.

remaining 34, 843 are of class CI. We have generated Unity Fingerprint descriptors [3], 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 1's.

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

- Azido pyrimidines,
- Natural products or antibiotics,
- Benzodiazepines, thiazolobenzimidazoles and related Compounds,
- Pyrimidine nucleosides,
- Dyes and polyanions,
- Heavy metal compounds, and
- Purine nucleosides.

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

We next generated Neighborgrams for classes CA and CM on $\frac{2}{3}$ of the data set and used the remaining $\frac{1}{3}$ for testing the generalization ability of the clustering method. Table 5 summarizes the results. Overall we recognized more than 50% of the confirmed actives and misclassified less than 2% of the confirmed inactives. An interesting observation was the relatively bad performance on the confirmed moderatively active compounds. Most of the clusters for class CM only cover one single training example. Therefore, it seems as if no real groups of compounds for this category exist in Unity-fingerprint space which would explain the bad generalization performance for this class.

However, we were more interested in the visual aspect of Neighborgrams for this application. Fig. 3 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

Table 5
Results on the training and test portion of the NCI AIDS Antiviral Screening data set (*Purity*=90%, triangular membership function)

Original label	Classified as					
	Training data			Test data		
	CA	CI	No cluster	CA	CI	No cluster
CA	195	0	10	65	12	43
CM	0	539	35	10	62	231
CI	7	0	23,231	46	223	11,336

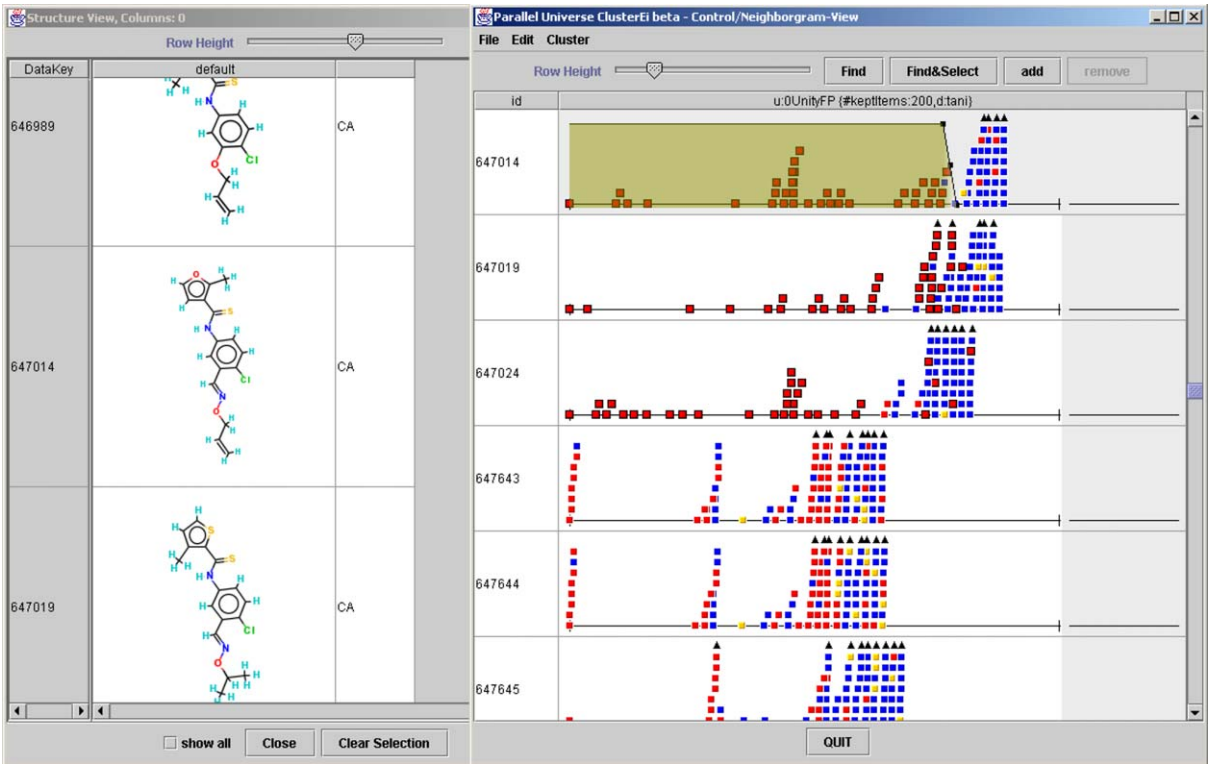


Fig. 3. The first cluster of the NIH-aids data centered around compound #647014. On the right the Neighborgrams in unity-fingerprint space (dark gray=CA, light gray=CM, black=CI), on the left a view showing some of the structures contained in this cluster.

cluster, both of them with slightly worse potential *Purity* and *Coverage*. At first we were surprised to see that none of the compounds contained in this cluster fall in any of the classes of active compounds listed on NIHs website [5]. 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 too toxic and hence not very interesting as active compounds to fight HIV. This is therefore a nice example of a

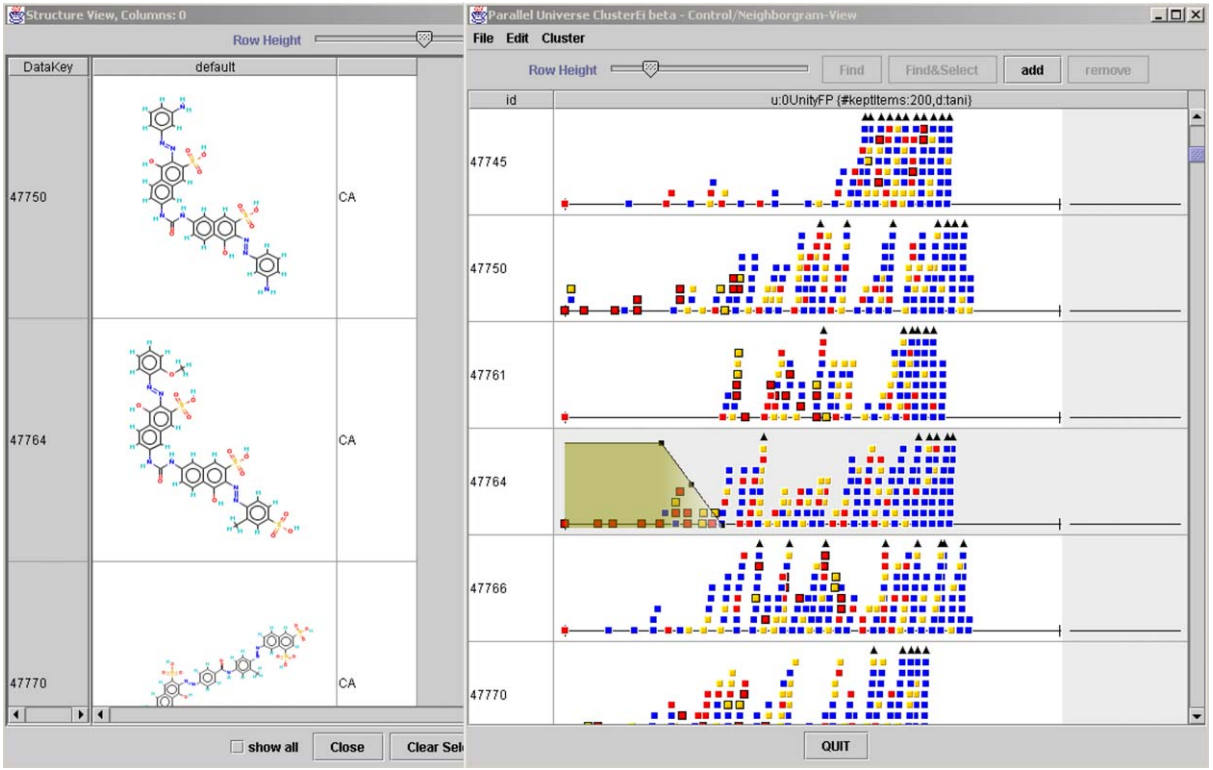


Fig. 4. Another cluster of the NIH-aids data centered around compound #47764 (right: Neighborgrams (dark gray=CA, light gray=CM, black=CI), left: structures). This cluster nicely covers one of the classes of active compounds: Dyes and Polyanions.

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 rank this first without any expert interaction.

Subsequent clusters reveal groupings very much in line with the classes listed above, one particular example is shown in Fig. 4. 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). Fig. 5 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 to quickly visually inspect promising cluster candidates. 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.

5. Discussion

A couple of issues that we do not have space to discuss in detail, but that are worth being touched upon at least briefly are listed in the following.

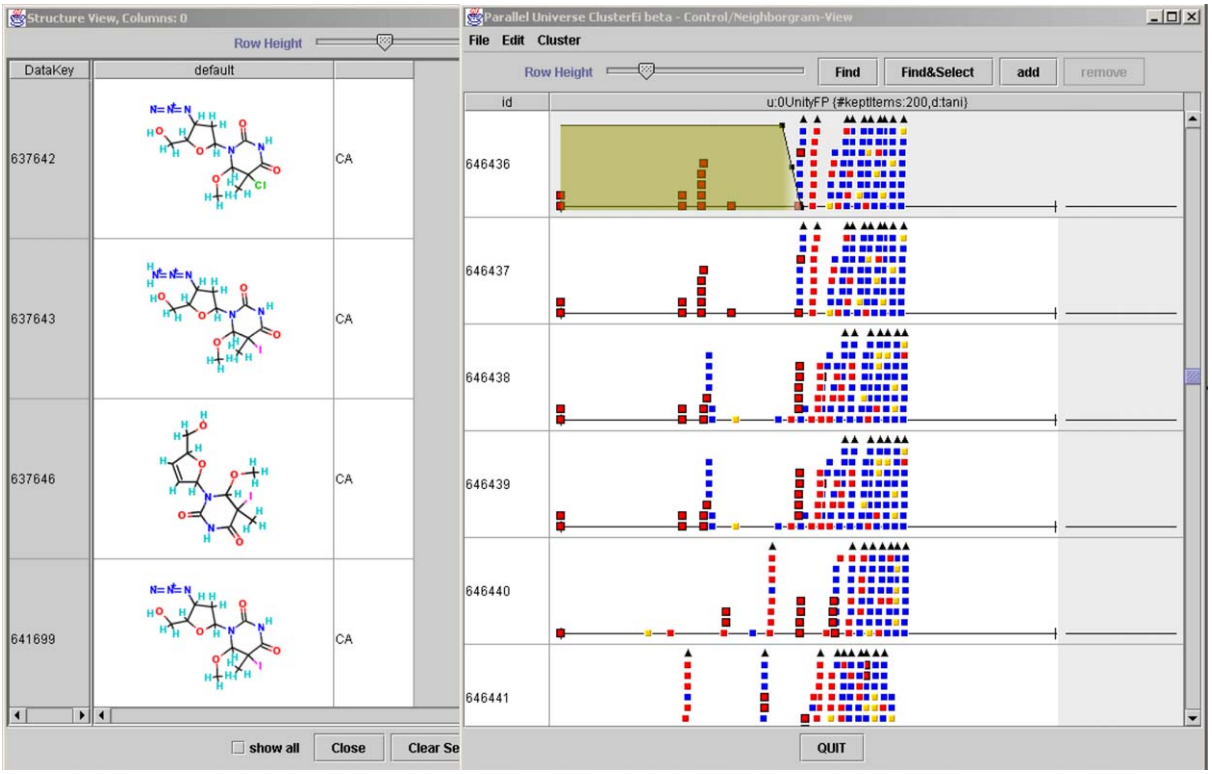


Fig. 5. Another cluster of the NIH-aids data centered around compound #646436 (right: Neighborgrams (dark gray=CA, light gray=CM, black=CI), left: structures). This cluster nicely covers part of one of the most well-known classes of active compounds: Azido Pyrimidines.

5.1. Binning Neighborgrams

Obviously for only few hundreds of patterns in each Neighborgram it is possible to plot all patterns individually. For larger neighborhoods it is preferable to bin the neighborhood and just display how many patterns of each class are encountered in each bin. We have experimented with this type of display as well but for all our applications smaller neighborhoods have shown to be sufficient to find good clusters. In domains centered around chemical descriptors this does not come as a surprise since larger neighborhoods have been shown to exhibit not much predictive ability in most cases anyway [8].

5.2. Fuzzy class membership

In some applications class information is not as exact as the examples or the presented benchmark data sets seem to suggest. Here fuzzifying the class information as well could allow to build better clusters. The purity of a cluster candidate would then be computed based on the degree of membership to the correct vs. conflicting class. We have not yet implemented this feature but believe that it offers promise for less perfect data, such as activity information as presented in the NCI HIV example above.

5.3. Parallel universes

Visual clustering and clustering using Neighborgrams are both very promising concepts. The Neighborgram approach in particular has one interesting extension that we are currently exploring. When looking at the algorithm in Table 1 one notes that the clusters don't "interact" directly. Besides the fact that a chosen cluster removes covered patterns from consideration there is no obvious need for two clusters to be based on the same distance function or even to originate from the same feature space. This leads to the notion of *Parallel Universes*, where we can find clusters in different feature spaces in parallel. Especially for data sets that involve structural descriptions of molecules it is hardly ever known which descriptor is optimal for a particular problem at hand. We can then modify the clustering algorithm to investigate all descriptor spaces in parallel and choose the best cluster among all universes in parallel. Covered patterns will subsequently be removed from all universes and the result is a set of clusters, spread out over different descriptor spaces.

5.4. Detecting outliers and mislabeled training instances

A problem that often occurs when collecting large amounts of biological data is the reliability of the labels. Most methods that are used to perform high throughput screening in drug discovery have a tendency to mislabel a substantial subset of the measured data (for example due to mechanical failures or unexpected side effects). The presented approach offers an interesting option to discover at least some of these wrong labels. By letting the user investigate "bad" Neighborgrams (i.e. Neighborgrams that have a high density of wrong patterns surrounding a center of different class), we can point out potential false positives. Preliminary experiments with this option using some real high throughput screening data were very encouraging.

6. Conclusions

We have presented a method to build clusters based on Neighborgrams, which model the local distribution of patterns for each potential cluster candidate. The underlying cluster algorithm has two main parameters: purity, which determines the boundary of a cluster candidate, and a termination criterion. Both parameters are easy to interpret and therefore not very critical to adjust.

We showed that the algorithm achieves satisfactory classification accuracy and that the introduction of fuzzy boundaries increases the performance of the cluster algorithm substantially, so that the results are comparable to other state of the art techniques.

The accompanying visualization technique provides means to explore the proposed cluster selection and enables the user to inject domain knowledge into the clustering process, as demonstrated using a bioinformatics application.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback on the first version of this article.

References

[1] M. Berthold, D.J. Hand (Eds.), Intelligent Data Analysis: An Introduction, second ed., Springer, Berlin, 2003.

[2] M.R. Berthold, J. Diamond, Constructive training of probabilistic neural networks, Neurocomputing 19 (1998) 167–183.

[3] R.D. Clark, Relative and absolute diversity analysis of combinatorial libraries, in: Combinatorial Library Design and Evaluation, Marcel Dekker, New York, 2001, pp. 337–362.

[4] F. Höppner, F. Klawonn, R. Kruse, T. Runkler, Fuzzy cluster analysis: methods for classification, Data Analysis and Image Recognition, Wiley, New York, 1999.

[5] http://dtp nci.nih.gov/docs/aids/aids_data html.

[6] D.A. Keim, Information visualization and visual data mining, IEEE Trans. Visualization Comput. Graphics 8 (1) (2002) 1–8.

[7] D. Michie, D.J. Spiegelhalter, C.C. Taylor (Eds.), Machine Learning, Neural and Statistical Classification, Ellis Horwood Limited, 1994.

[8] D. Patterson, R. Cramer, A. Ferguson, R. Clark, L. Weinberger, Neighborhood behavior: a useful concept for validation of molecular diversity descriptors, J. Med. Chem. 39 (1996) 3049–3059.

[9] D.F. Specht, Probabilistic neural networks, in: Neural Networks, vol. 3, 1990, pp. 109–118.

[10] O. Weislow, R. Kiser, D. Fine, J. Bader, R. Shoemaker, M. Boyd, New soluble formazan assay for HIV-1 cytopathic effects: application to high flux screening of synthetic and natural products for AIDS antiviral activity, J. Natl. Cancer Inst. 81 (1989) 577–586.

[11] R.R. Yager, D.P. Filev, Approximate clustering via the mountain method, IEEE Trans. Systems, Man Cybernet. 24 (8) (1994) 1279–1284.