First publ. in: Proceedings: 2002 IEEE International Conference on Data Mining ICDM 2002, 9-12 December 2002, Maebashi City, Japan / Ed.: Vipin Kumar ... pp. 581-584

Neighborgram Clustering
Interactive Exploration of Cluster Neighborhoods

Michael R. Berthold, Bernd Wiswedel, and David E. Patterson
Data Analysis Research Lab, Tripos Inc.
601 Gateway Blvd., Suite 720; South San Francisco, CA 94080; USA
email: {berthold,bwiswede,pat}@tripos.com

Abstract

We describe an interactive way to generate a set of clusters for a given data set. The clustering is done by constructing local histograms, which can then be used to visualize, select, and fine-tune potential cluster candidates. The accompanying algorithm can also generate clusters automatically, allowing for an automatic or semi-automatic clustering process where the user only occasionally interacts with the algorithm. We illustrate the ability to automatically identify and visualize clusters using NCI's AIDS Antiviral Screen data set.

1. Introduction

The analysis of large data sets usually results in the extraction of models that describe some aspect of the process that originally generated the data. In many real world applications, users are often willing to accept models with not-optimal generalization performance if they can explore the underlying decision process and, preferably, are able to influence the construction of the model interactively throughout the training process. A summary of methods for interactive visualization can be found in [7].

This paper presents an interactive visualization technique for a clustering algorithm, which provides such an interactive model-construction method. The presented method visualizes cluster neighborhoods in detail by displaying each example in the clusters' neighborhood individually and is accompanied by a clustering algorithm, which finds close-to-optimal cluster centers for certain classes of interest. The visualization component allows the user to interact with the clustering algorithm, thus inserting domain knowledge into the cluster formation process. In addition, the clustering process is not limited to low dimensional feature space, in fact we have successfully used the presented approach in feature spaces with thousands of features.

The algorithm described here belongs to the category of clustering techniques which pick representative examples from the training data (similar to the algorithm described in [3]) rather than represent prototypes by weighted averages of several training points (such as for example fuzzy c-means [5]). Instead of relying on the usual heuristics or greedy algorithms to select example patterns as cluster representatives, the presented method analyses the neighborhood of each cluster candidate and picks the optimal cluster representative directly. This neighborhood can additionally be visualized to give the user insights into the patterns each cluster candidate represents. Such a complete and hence computationally expensive approach obviously only works for all classes of a medium sized data set or - in case of very large data sets - to model a minority class of interest. In many real world applications this scenario is the one that matters, however. Especially in bioinformatics applications it is often more important to extract detailed knowledge about few, rare classes within the larger context.

2. Neighborgram Clustering

In the following we assume a feature space in which M training instances $\vec{x}_i$ are given ($i = 1 \ldots M$) along with a function $d(\cdot)$, which computes distances between training instances. Each example $\vec{x}_i$ is also assigned to one of C classes, indicated by the class index k_i ($k_i = 1 \ldots C$).

2.1. Neighborgrams

The basic algorithm operates on all training examples in parallel and computes a so-called Neighborgram for each example of the class(es) of interest. A Neighborgram records the patterns and associated classes of the immediate neighbors for the corresponding center $\vec{x}_i$ in an ordered list:

$$\mathrm{NG}_i = [(l_1, k_1), \ldots, (l_r, k_r), \ldots, (l_R, k_R)]$$

where $l_r = 1 \ldots M$ indicates the index of a training example and $k_r = 1 \ldots C$ is the corresponding class index. The list is

sorted according to the distance of pattern $\vec{x}_{l_r}$ to the center vector $\vec{x}_i$:

$$\forall r \ : \ 2 \leq r \leq R \wedge d(\vec{x}_i, \vec{x}_{l_{(r-1)}}) \leq d(\vec{x}_i, \vec{x}_{l_r}).$$

Note that $l_1 = i$, because $d(\vec{x}_i, \vec{x}_i) = 0$ for all i, that is, each pattern is closest to itself. The overall length of this list is determined by parameter R, where $R \ll M$ for large data sets. Hence a Neighborgram simply lists all neighbors of a particular pattern in order of their distance, up to a certain depth of the list.

Obviously in case of large data sets the computation of Neighborgrams for each training pattern is excessively time and memory consuming. As noted earlier, however, 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 $O(M \cdot M') \cdot O(d(\cdot))$, where $O(d(\cdot))$ depends linearly on the dimension of the feature space for most distance functions and M' indicates the number of examples of the minority class(es), i.e. $M' \ll M$ in case of large data sets.

2.2. The Basic Clustering Algorithm

The main idea behind the clustering algorithm based on Neighborgrams can be summarized as follows: determine cluster-candidates from each Neighborgram, then find the "best" Cluster, and remove all patterns it "covers". Now find the next "best" Cluster, remove all patterns that are covered and so on. Obviously the notions of "best" and "covers" need to be clarified, and we need to explain how a suitable cluster candidate can be derived from each Neighborgram. In order to do this, we first introduce a number of measures:

- *Purity*: The purity of a Cluster is computed based on the Neighborgram i it stems from. Purity basically determines how many patterns of the correct class are contained within a certain neighborhood of depth r with respect to patterns of all classes inside this area:

$$\mathrm{Purity}_i(r) = \frac{|\,\{(l_{r'}, k_{r'}) \in \mathrm{NG}_i \mid 1 \leq r' \leq r \wedge k_{r'} = k_i\}\,|}{|\,\{(l_{r'}, k_{r'}) \in \mathrm{NG}_i \mid 1 \leq r' \leq r\}\,|}$$

- *OptDepth*: is the optimal depth for which a certain $\mathrm{Purity} = p_{\min}$ is guaranteed, that is,

$$\mathrm{OptDepth}_i(p_{\min}) = \max \mathrm{arg}_r \{ \mathrm{Purity}_i(r) \geq p_{\min} \wedge \mathrm{Purity}_i(r+1) < p_{\min}\}$$

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

$$\mathrm{Coverage}_i(r) = |\,\{(l_{r'}, k_{r'}) \in \mathrm{NG}_i \mid 1 \leq r' \leq r \wedge k_i = k_{r'}\}\,|$$

We can now specify clearer what we mean by "best" cluster and "covers". Starting from a user-defined value for parameter *Purity* we can compute values for parameters *OptDepth* and *Coverage* for each Cluster. The best cluster is then the one with the highest *Coverage*. This cluster "covers" all patterns that are within its radius.

The only remaining parameter is a limit on the overall amount of coverage desired: *MaxCoverage*. Once the sum of all covered patterns exceeds this threshold, the algorithm terminates. The following pseudo code summarizes the resulting algorithm:

```
1)  ∀x_i: k_i is minority class ⇒ compute NG_i
2)  ∀NG_i: compute OptDepth_i
3)  ∀NG_i: compute Coverage_i
4)  while MaxCoverage not reached:
5)     i_best = max arg_i{Coverage_i(OptDepth_i)}
6)     add (i_best, OptDepth_{i_best}) to list of clusters
7)     determine list of covered patterns
8)     remove them from all Neighborgrams NG_i
9)     ∀NG_i: recompute Coverage_i
10) end while
```

3. Experimental Results and Visual Clustering

In the following we will focus on the main contribution of the presented method, the ability to visually investigate cluster candidates. Obviously the presented algorithm could also be used as a stand-alone clustering method. The resulting classification performance is comparable to state-of-the-art classification methods and performs on par with the method presented in [3].

3.1. Neighborgram Visualization

Visualizing a Neighborgram requires only one dimension, since we are interested in the distance to the center point only. In addition, we are usually only interested in a small neighborhood (i.e. Neighborgrams with a small depth R) and can invest some screen area for each individual neighbor. Figure 1 shows an example of a visualization of one Neighborgram built for a pattern of class Iris-Setosa (x_{l_1}). In case that two or more patterns are too close to each other so that they would overlap we decided to stack them on top of each other. The vertical axes therefore has no geometrical meaning, it is simply used to avoid overlaps[1].

[1] Obviously many other ways to depict one-dimensional spaces can be used, dense-pixel displays [6] come to mind if the neighborhood to be displayed contains several thousands or more patterns.

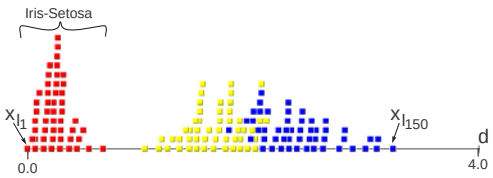

Figure 1. A Neighborgram for the Iris data.

Note how in this case, all 50 patterns of class Iris-Setosa are in a close neighborhood of x_{l_1}, and the two other classes are clumped together further apart. In this case the depth was chosen to be $R = 150$, i.e. the last pattern shown is $x_{l_{150}}$. This particular example shows a good cluster candidate, which is also the one returned first by the algorithm explained above.

As a test case we used a well-known data set from the National Cancer Institute, the DTP AIDS Antiviral Screen data set [8]. We have used the class assignment provided with the data, that is, 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**), and compounds not meeting these criteria were listed as confirmed inactive (**CI**). Available online [1] are screening results and chemical structural data on compounds. We have generated Unity Fingerprint descriptors [4], which represent each compound through a 990-dimensional bit string. Unity fingerprints represent a collection of pre-defined chemical substructures of interest. The used distance metric was the usual 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 one or both vectors.

We next generated Neighborgrams for classes **CA** and **CM**. The first and biggest cluster covered quite a large number of patterns of class **CA**. 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 NIH's website [1]. 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 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 known classes of compounds, one particular example is shown in Figure 2. Here the group of Azido Pyrimidines is rediscovered, probably one of the best-known class of active compounds for HIV.

Experiments with this (and other similar) data sets showed nicely how the interactive clustering using Neighborgrams helps to include domain knowledge in the clustering process and how Neighborgrams help to quickly display 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.

4 Extensions

A couple of issues that we do not have space to discuss in detail, but that are worth being mentioned are listed in the following:

Partial Coverage and Fuzzy Clusters: It is obvious that the basic algorithm sketched in the previous section 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 introduced the notion of *Partial Coverage* using fuzzy membership functions [9], which allow us to model a degree of membership of a particular pattern to a cluster.

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.

Fuzzy Class Membership: In many pharmaceutical applications class information is not as exact as the example above seems 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.

Minimum Cluster Size: 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 encountering of a pattern of wrong class will set *OptDepth* very close to the cluster's center. To avoid this, we have introduced a parameter *minSize*, which allows to specify a minimum number of patterns in a neighborhood before *Purity* and *OptDepth* are determined. Early experiments with noisy data sets have shown a decrease in number of clusters and better generalization ability.

Parallel Universes: The algorithm described above does not require that the clusters reside in the same feature space. 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 the same features. Hence we can find clusters in different fea-

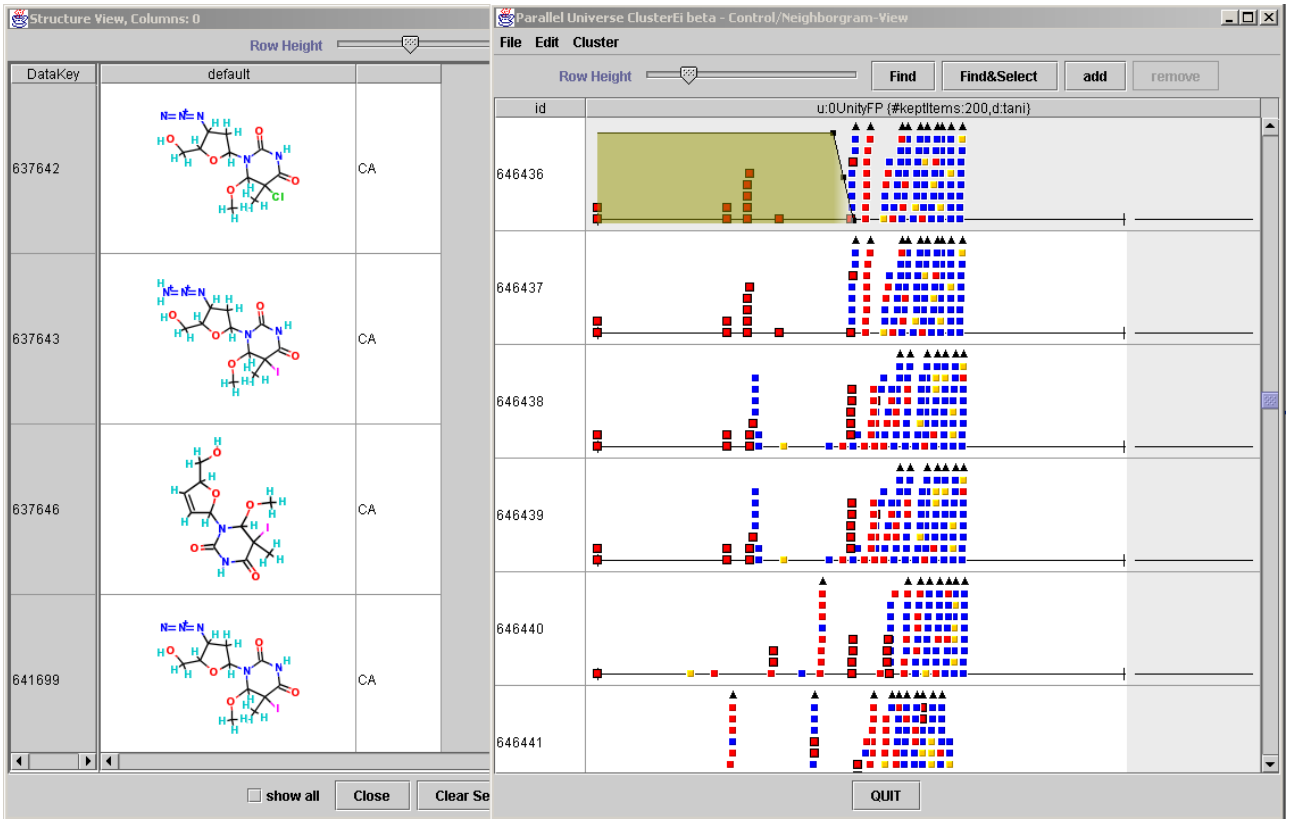

Figure 2. One cluster for the NIH-Aids data (top row) covering Azido Pyrimidines.

ture spaces in parallel. Covered patterns will then be removed from all universes and the result is a set of clusters, spread out over different descriptor spaces.

Detecting Outliers and Mislabeled Training Instances: A problem that often occurs when collecting large amounts of biological data is the reliability of the labels. 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.

5. 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 proposed visualization of Neighborgrams allows the user to explore the proposed cluster selection and enables experts to inject domain knowledge into the clustering process by selecting, discarding, or fine-tuning cluster candidates. Analysis of chemical data has shown the usefulness of the provided visualization. By displaying cluster candidates it is possible to naturally incorporate expert knowledge through interactive feedback.

References

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

[2] M. Berthold and D. J. Hand, editors. *Intelligent Data Analysis: An Introduction*. Springer Verlag, second edition, 2002.

[3] M. R. Berthold and J. Diamond. Constructive training of probabilistic neural networks. *Neurocomputing*, 19:167–183, 1998.

[4] R. D. Clark. Relative and absolute diversity analysis of combinatorial libraries. In *Combinatorial Library Design and Evaluation*, pages 337–362. Marcel Dekker, New York, 2001.

[5] R. Davé and R. Krishnapuram. Robust clustering methods: A unified view. *IEEE Transactions on Fuzzy Systems*, 5(2):270–293, May 1997.

[6] D. A. Keim and H.-P. Kriegel. Visualization techniques for mining large databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):923–938, 1996.

[7] D. A. Keim and M. Ward. Visualization. In *[2, Chapter 11]*.

[8] O. Weislow, R. Kiser, D. Fine, J. Bader, R. Shoemaker, and 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. *Journal National Cancer Institute*, 81:577–586, 1989.

[9] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.