Ersch. in: Advances in intelligent data analysis X : 10th international symposium, IDA 2011, Porto, Portugal, October 29 - 31, 2011 ; proceedings / João Gama... (eds.). - Heidelberg [u.a.] : Springer, 2011. - S. 388-400. - (Lecture notes in computer science ; 7014). - ISBN 978-3-642-24799-6

http://dx.doi.org/10.1007/978-3-642-24800-9_36

Supervised Learning in Parallel Universes Using Neighborgrams

Bernd Wiswedel and Michael R. Berthold

Department of Computer and Information Science, University of Konstanz, Germany

Abstract. We present a supervised method for *Learning in Parallel Universes*, i.e. problems given in multiple descriptor spaces. The goal is the construction of local models in individual universes and their fusion to a superior global model that comprises all the available information from the given universes. We employ a predictive clustering approach using *Neighborgrams*, a one-dimensional data structure for the neighborhood of a single object in a universe. We also present an intuitive visualization, which allows for interactive model construction and visual comparison of cluster neighborhoods across universes.

1 Introduction

Computer-driven learning techniques are based on a suitable data-representation of the objects being analyzed. The classical concepts and techniques that are typically applied are all based on the assumption of one appropriate unique representation. This representation, typically a vector of numeric or nominal attributes, is assumed to sufficiently describe the underlying objects. In many application domains, however, various different descriptions for an object are available. These different descriptor spaces for the same object domain typically reflect different characteristics of the underlying objects and as such often even have their own, unique semantics and can and should therefore not be merged into one descriptor.

As most classical learning techniques are restricted to learning in exactly one descriptor space, the learning in the presence of several object representations is in practice often solved by either reducing the analysis to one descriptor, ignoring the others; by constructing a joint descriptor space (which is often impossible); or by performing independent analyses on each space. All these strategies have limitations because they either ignore or obscure the multiple facets of the objects (given by the descriptor spaces) or do not respect overlaps. This often makes them inappropriate for practical problems.

As a result *Learning in Parallel Universes* [13] has emerged as a novel learning scheme that encompasses the simultaneous analysis of all given descriptor spaces, i.e. universes. It deals with the concurrent generation of local models for each universe, whereby these models cover local structures that are unique for individual universes and at the same time can also cover other structures that span multiple universes. The resulting global model outperforms the above mentioned schemes and often also provides new insights with regard to overlapping as well as universe-specific structures. Although the learning task itself can be both supervised (e.g. building a classifier) and

Konstanzer Online-Publikations-System (KOPS)
URN: http://nbn-resolving.de/urn:nbn:de:bsz:352-193242

unsupervised (e.g. clustering), we concentrate in the following on supervised problem scenarios.

In this paper we discuss an extension to the *Neighborgram* algorithm [4,5] for learning in parallel universes. It is a supervised learning method that has been designed to model small or medium sized data sets or to model a (set of) minority class(es). The latter is commonly encountered in the context of activity prediction for drug discovery. The key idea is to represent an object's neighborhoods, which are given by the various similarity measures in the different universes, by using neighborhood diagrams (so-called neighborgrams). The learning algorithm derives cluster candidates from each neighborgram and ranks these based on their qualities (e.g. coverage). The model construction is carried out in a sequential covering-like manner, i.e. starting with all neighborgrams and their cluster candidates, taking the numerically best one, adding it as cluster and proceeding with the remaining neighborgrams while ignoring the already covered objects. We will present and discuss extensions to this algorithm that reward clusters which group in different universes simultaneously, and thus respect overlaps. As we will see this can significantly improve the classification accuracy over just considering the best cluster at a time.

Another interesting usage scenario of the neighborgram data structure is the possibility of displaying them and thus involving the user in the learning process. Using the visualization techniques described in [4] in a grid view, which aligns the different universes column by column, the user can inspect the different neighborhoods across the available universes and assess possible structural overlaps.

2 Related Work

Subspace Clustering. Subspace clustering methods [11] operate on a single, typically very high-dimensional input space. The goal is to identify regions of the input space (a set of features), which exhibit a high similarity on a subset of the objects, or, more precisely, the objects' data representations. Subspace clustering methods are commonly categorized into bottom-up and top-down approaches. Bottom-up starts by considering density on individual features and then subsequently merging features/subspaces to subspaces of higher dimensionality, which still retain a high density. The most prominent example is CLIQUE [2], which partitions the input space using a static grid and then merges grid elements if they meet a given density constraint. Top-down techniques initially consider the entire feature space and remove irrelevant features from clusters to compact the resulting subspace clusters. One example in this category is COSA [10], a general framework for this type of subspace clustering; it uses weights to encode the importance of features to subspace clusters. These weights influence the distance calculation and are optimized as part of the clustering procedure. Subspace clustering methods share with learning in parallel universes that they try to respect locality also in terms of the features space, although they do it on a different scale. They refer to individual features whereas we consider universes, i.e. semantically meaningful descriptor spaces, which are given a-priori.

Multi-View-Learning. In multi-view-learning [12] one assumes a similar setup as in learning in parallel universes, i.e. the availability of multiple descriptor spaces

(universes or *views*). This learning concept has a different learning scope since it expects all universes/views to share the same structure. Therefore each individual universe would suffice for learning if enough training data were available. One of the first works in the multi-view area was done by Blum and Mitchell [6], who concentrate on the classification of web pages. They introduce *co-training* in a semi-supervised setting, i.e. they have a relatively small set of labeled and a rather large set of unlabeled instances. The web pages are described using two different views, one based on the actual content of the web site (bag of words), the other one based on anchor texts of hyperlinks pointing to that site – both views are assumed to be *conditionally independent*. Co-training creates a model on each view, whereby the training set is augmented with data that was labeled with high confidence by the model of the respectively other view. This way the two classifiers bootstrap each other. There is also a theoretical foundation for the co-training setup: It was shown that the disagreement rate between two independent views is an upper bound for the error rate of either hypothesis [9]. This principle already highlights the identical structure property of all views as the base assumption of multi-view learners, which is contrary to the setup of learning in parallel universes, where some information may only be available in a subset of universes.

There exist a number of other similar learning setups. These include ensemble learning, multi-instance-learning and sensor fusion. We do not discuss these methods in detail but all have either a different learning input (no a-priori defined universes) or a distinct learning focus (no partially overlapping structures across universes while retaining universe semantics).

3 Learning in Parallel Universes

Learning in parallel universes refers to the problem setting of multiple descriptor spaces, whereby single descriptors are not sufficient for learning. Instead we assume that the information is distributed among the available universes, i.e. individual universes may only explain a part of the data. We propose a learning concept, which overcomes the limitations outlined above by a simultaneous analysis of all available universes. The learning objective is twofold. First, we aim to identify structures that occur in only one or few universes (for instance groups of objects that cluster well in one universe but not in others). Secondly we want to detect and facilitate overlapping structures between multiple, not necessarily all, universes (for instance clusters that group in multiple universes). The first aim addresses the fact that a universe is not a complete representation of the underlying objects, that is, it does not suffice for learning. The task of descriptor generation is in many application domains a science by itself, whereby single descriptors (i.e. universes) carry a semantic and mirror only certain properties of the objects. The second aim describes cases, in which structures overlap across different universes. For clustering tasks, for instance, this would translate to the identification of groups of objects that are self-similar in a subset of the universes. Note that in order to detect these overlaps and to support their formation it is necessary for information to be exchanged between the individual models during their construction. This is a major characteristic of learning in parallel universes, which cannot be realized with any of the other schemes outlined in the previous section.

3.1 Application Scenarios

The challenge of learning in parallel universes can be found in almost all applications that deal with the analysis of complex objects. We outline a few of these below.

Molecular data. A typical example is the activity prediction of drug candidates, typically small molecules. Molecules can be described in various ways, which potentially focus on different aspects [3]. It can be as simple as a vector of numerical properties such as molecular weight or number of rotatable bonds. Other descriptors concentrate on more specific properties such as charge distribution, 3D conformation or structural information. Apart from such semantic differences, descriptors can also be of a different type including vector of scalars, chemical fingerprints (bit vectors) or graph representations. These diverse representations make it impossible to simply join the descriptors and construct a joint feature space.

3D object data. Another interesting application is the mining of 3D objects [8]. The literature lists three main categories of descriptors: (1) image-based (features describing projections of the object, e.g. silhouettes and contours), (2) shape-based (like curvature measures) and (3) volume-based (partitioning the space into small volume elements and then considering elements that are completely occupied by the object or contain parts of its surface area, for instance). Which of these descriptions is appropriate for learning, mostly depends on the class of object being considered. For instance image- or volume-based descriptors fail on modeling a class of humans, taking different poses, since their projections or volumes differ, whereas shape-based descriptors have proven to cover this class nicely.

Image data. There are also manifold techniques available to describe images. Descriptors can be based on properties of the color or gray value distribution; they can encode texture information or properties of the edges. Other universes can reflect user annotations like titles that are associated with the image. Also in this domain it depends sometimes on the descriptor, whether two images are similar or not.

4 Neighborgrams in Parallel Universes

In the following we describe the neighborgram data structure [4], discuss the fully automated clustering algorithm and highlight the advantages of the visualization. We consider a set of objects i, each object being described in U, $1 \leq u \leq U$, parallel universes. The object representation for object i in universe u is $x_{i,u}$. Each object is assigned a class $c(i)$. We further assume appropriate definitions of distance functions for each universe $d^{(u)}(x_{i,u}, x_{j,u})$.

4.1 Neighborgram Data Structure

We define a neighborgram as a list of R-nearest neighbors to an object i in a universe u:

$$\mathrm{NG}_i^{(u)} = \left\langle x_{l_{i,1}^{(u)},u}, \ldots, x_{l_{i,R}^{(u)},u} \right\rangle .$$

The subscript $l_{i,r}^{(u)}$ reflects the ordering of the neighbors according to the distance to the centroid. For the sake of simplicity we abbreviate $l_{i,r}^{(u)}$ with l_r and also omit the second subscript u, always accounting for the fact that the list contains objects referring to a centroid object i in a specific universe u. The ordering implies for any neighborgram $l_{i,1}^{(u)} = l_1 = i$ since the centroid is closest to itself. This list is not necessarily a unique representation of the neighborhood as objects may have an equal distance to the centroid. However, this is not a limitation to the algorithm.

Neighborgrams are constructed for all objects of interest, i.e. either for all objects of a small- or medium-size data set or the objects belonging to one or more target classes, in all universes.

As an illustrative example consider the data set in figure 1, which is given in a single universe. This universe is 2-dimensional (shown on the left) with two different classes (empty and filled circles). The neighborgrams for the three numbered empty objects are shown on the right. We use the Manhattan norm as distance function, i.e. the distance between two objects is the number of steps on the grid (for instance the distance between ① and ③ is 3, two steps in horizontal and one in vertical direction). The corresponding list representation is then

$$\mathrm{NG}_{①} = \langle ①, ●, ②, ③, ● \rangle$$
$$\mathrm{NG}_{②} = \langle ②, ③, ①, ●, ○ \rangle$$
$$\mathrm{NG}_{③} = \langle ③, ②, ●, ①, ○ \rangle$$

We will use these neighborgrams in the following to introduce basic measures that help us to derive cluster candidates and to determine numeric quality measures.

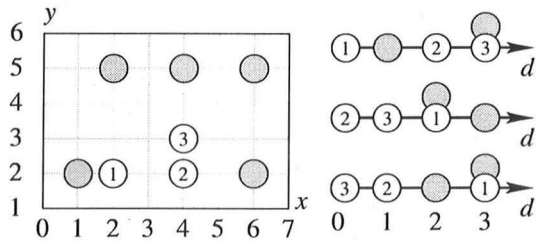

Fig. 1. Sample input space with neighborgrams for the objects ①, ② and ③

4.2 Neighborgram Clustering Algorithm

The basic idea of the learning algorithm is to consider each neighborgram in each universe as a potential (labeled) cluster candidate. These candidates are derived from the list representation, are based on the class distribution around the centroid, and are assigned the class of their respective centroids. Good clusters will have a high *coverage*, i.e. many objects of the same class as the centroid in the close vicinity, whereby cluster boundaries are represented by the distance of the farthest neighbor satisfying some purity constraint. The basic clustering algorithm will iteratively add the cluster candidate with the highest coverage to the set of accepted clusters, remove the covered objects

from consideration, and start over by re-computing the coverage of the remaining candidates. The interesting point here is that this basic algorithm is not restricted to learn in a single universe. In fact, this basic algorithm is free to choose the best cluster candidate from all universes, thereby directly building a cluster set with cluster from different origins and hence a model for parallel universes.

Before we provide the algorithm in pseudo-code let us define some measures that are used to define a cluster candidate and its numerical quality. Each of the following values is assigned to a single neighborgram (but computed for all):

- *Coverage* $\Gamma_i^{(u)}$: The coverage describes how many objects of the same class as the centroid $c(i)$ are within a certain length r in the neighborgram for object i in universe u. These objects are covered by the neighborgram.

$$\Gamma_i^{(u)}(r) = \left|\left\{x_{l_{r'}} \in \mathrm{NG}_i^{(u)} \mid 1 \leq r' \leq r \wedge c(l_{r'}) = c(i)\right\}\right|$$

  For example, the coverages for $\mathrm{NG}_{①}$ in figure 1 are: $\Gamma_{①}(1) = 1$, $\Gamma_{①}(2) = 1$, $\Gamma_{①}(3) = 2$ and $\Gamma_{①}(4) = 3$.
- *Purity* $\Pi_i^{(u)}(r)$: The purity denotes the ratio of objects of the correct class (i.e. same class as the centroid) to all objects that are contained within a certain neighborhood of length r:

$$\Pi_i^{(u)}(r) = \frac{\left|\left\{x_{l_{r'}} \in \mathrm{NG}_i^{(u)} \mid 1 \leq r' \leq r \wedge c(l_{r'}) = c(i)\right\}\right|}{\left|\left\{x_{l_{r'}} \in \mathrm{NG}_i^{(u)} \mid 1 \leq r' \leq r\right\}\right|}.$$

  For instance, object ① in figure 1 would have purities: $\Pi_{①}(1) = 1$, $\Pi_{①}(2) = \frac{1}{2}$, $\Pi_{①}(3) = \frac{2}{3}$ and $\Pi_{①}(5) = \frac{3}{5}$.
- *Optimal Length* $\Lambda_i^{(u)}$: The optimal length is the maximum length for which a given purity threshold $p_{\min}$ is still valid. In practical applications it has shown to be reasonable to constrain the optimal length to be at least some minimum length value $r_{\min}$ in order to avoid clusters that cover only few objects.

$$\Lambda_i^{(u)}(p_{\min}, r_{\min}) = \max\left\{r \mid r_{\min} \leq r' \leq r \wedge \Pi_i^{(u)}(r') \geq p_{\min}\right\}.$$

  Note, this term may be undefined and practically often is for noisy data. In the example in figure 1 the optimal length values are for a minimum length $r_{\min} = 1$ (unconstrained): $\Lambda_{②}(1,1) = 2$ and $\Lambda_{②}(\frac{2}{3},1) = 3$.

Apart from the specification of a minimum purity $p_{\min}$ and a minimum cluster size $r_{\min}$, the user has to specify a minimum overall coverage $\psi_{\min}$. If the sum of coverage values of all accepted clusters is greater than this value, the algorithm terminates. The basic algorithm is outlined in algorithm 1. The initialization starts with the construction of neighborgrams for all objects of certain target classes (minority class(es) or all classes) in line 1. Prior to the learning process itself, it determines a cluster candidate for each

Algorithm 1. Basic Neighborgram Clustering Algorithm

1: $\forall i,u$: $c(i)$ is target class $\Rightarrow$ construct $\mathrm{NG}_i^{(u)}$
2: $\forall \mathrm{NG}_i^{(u)}$: compute $\Lambda_i^{(u)} = \Lambda_i^{(u)}(p_{\min}, r_{\min})$
3: $s \leftarrow 0$ /* accumulated coverage */
4: $\mathcal{NG} \leftarrow \emptyset$ /* (empty) result set */
5: $\hat{i} \leftarrow$ undefined /* current best Neighborgram */
6: **repeat**
7: $\quad \forall \mathrm{NG}_i^{(u)}$: compute $\Gamma_i^{(u)}\left(\Lambda_i^{(u)}\right)$
8: $\quad (\hat{i},\hat{u}) \leftarrow \arg\max_{(i,u)} \left\{\Gamma_i^{(u)}\left(\Lambda_i^{(u)}\right)\right\}$
9: $\quad$ **if** $\hat{i}$ is undefined **then**
10: $\quad\quad \hat{\omega} \leftarrow \Lambda_{\hat{i}}^{(\hat{u})}$ /* optimal length best Neighborgram */
11: $\quad\quad M_{\text{covered}} \leftarrow \left\{ l_{\hat{i},r'}^{(\hat{u})} \mid 1 \le r' \le \hat{\omega} \wedge c\left(l_{\hat{i},r'}^{(\hat{u})}\right) = c\left(\hat{i}\right)\right\}$
12: $\quad\quad \forall \mathrm{NG}_i^{(u)}$: $\mathrm{NG}_i^{(u)} \leftarrow \mathrm{NG}_i^{(u)} \setminus \{x_{j,u} \mid j \in M_{\text{covered}}\}$
13: $\quad\quad s \leftarrow s + \Gamma_{\hat{i}}^{(\hat{u})}(\hat{\omega})$
14: $\quad\quad \mathcal{NG} \leftarrow \mathcal{NG} \cup \left\{\left(\mathrm{NG}_{\hat{i}}^{(\hat{u})}, \hat{d}\right)\right\}$
15: $\quad$ **end if**
16: **until** $(s \ge \psi) \vee (\hat{i}$ is undefined$)$
17: **return** $\mathcal{NG}$

neighborgram, which is based on user defined values for the minimum purity and the minimum size (line 2). The learning phase is shown in line 6 to 16: It iteratively adds the cluster candidate with the highest (remaining) coverage (line 8) to the cluster set $\mathcal{NG}$, while removing the covered objects from all remaining neighborgrams. The algorithm terminates if either no more cluster candidates satisfy the search constraints (miss the minimum size constraint) or the accumulated coverage is larger than the required coverage $\psi_{\min}$.

This basic algorithm learns a set of clusters distributed over different universes, whereby each cluster is associated with a class (the class of its centroid). The prediction of unlabeled data is carried out by a best-matching approach, i.e. by identifying the cluster that covers the query object best. In case more than one cluster covers the query and these clusters are assigned different classes, the final class can be determined using a majority vote of the clusters.

Universe Interaction. The basic algorithm inherently enables simultaneous learning in different universes by considering all neighborgrams as potential cluster candidates and removing covered objects in all universes. However, this is a rather weak interaction as it does not respect overlaps between universes. These are often of special interest for mainly two reasons: Firstly, they give new insights regarding recurrent structures in different universes and help the expert to better understand relations between universes. Secondly, they may help to build a more robust model. The basic algorithm described above penalizes such overlapping structures since it removes covered objects from all remaining neighborgrams. If two clusters group equally in two universes, the algorithm would choose any one of the two clusters and remove the objects. The non-accepted

cluster will never be considered as a good cluster candidate again, because all its supporting objects were removed already.

In order to take these overlaps into account, we modify the algorithm to detect overlapping clusters during the clustering process. More formally, we define the relative overlap $\upsilon : \mathrm{NG} \times \mathrm{NG} \mapsto [0,1]$ of two neighborgrams as

$$\upsilon\left(\mathrm{NG}_i^{(u)}, \mathrm{NG}_j^{(y)}\right) = \frac{\left|M_i^{(u)} \cap M_j^{(y)}\right|}{\left|M_i^{(u)} \cup M_j^{(y)}\right|},$$

whereby $M_i^{(u)}$ denotes the set of objects of the same class as the centroid that are covered by the cluster candidate derived from the Neighborgram $\mathrm{NG}_i^{(u)}$ according to the user settings of minimum purity and size. This overlap is the cardinality of the intersection of these two sets divided by the size of the union. Two neighborgrams in different universes representing the same cluster will therefore have a high overlap.

Using the above formula we can numerically express the overlap between cluster candidates. We modify the basic algorithm to also add those candidates to the cluster set that have the highest overlap with the currently best cluster candidate in each universe unless their overlap is below a given threshold $\upsilon_{\min}$. That is, in each iteration (refer to line 6–16 in algorithm 1) we determine the cluster with the highest remaining coverage in all universes. Before removing the covered objects from further consideration we identify the cluster candidates in all remaining (i.e. non-winner) universes with the highest overlap wih respect to the current winner. For each of these overlapping cluster candidates we test whether the overlap is at least $\upsilon_{\min}$ and, if so, also accept it as cluster. This strategy of course increases the number of clusters in the final model but it also improves the prediction quality on test data considerably as we will see later.

Weighted Coverage. Another beneficial modification to the algorithm is the usage of a different cluster representation by means of gradual degrees of coverage. In the previous sections we used sharp boundaries to describe a cluster. An object is either covered by a cluster or not, independent of its distance to the cluster centroid. By using a weighted coverage scheme that assigns high coverage ($\rightarrow 1$) to the objects in the cluster center and low values ($\rightarrow 0$) to those at the cluster boundaries, we have a much more natural cluster representation [5]. As weighting scheme we use in the following a linear function; the degree of coverage $\mu_i^{(u)}(j)$ for an object j to the cluster candidate for a neighborgram $\mathrm{NG}_i^{(u)}$ with its centroid $x_{i,u}$ in universe u is:

$$\mu_i^{(u)}(j) = \begin{cases} \frac{\hat{d} - d^{(u)}\left(x_{i,u}, x_{j,u}\right)}{\hat{d}} & \text{if } d^{(u)}\left(x_{i,u}, x_{j,u}\right) \leq \hat{d} \\ 0 & \text{else,} \end{cases}$$

whereby $\hat{d}$ represents the distance of the farthest object covered by the neighborgram according to the optimal length $\Lambda_i^{(u)}$. Note that due to the gradual coverage the algorithm needs to be slightly adapted: The coverage of a cluster is no longer a simple count of the contained objects but an accumulation of their weighted coverage. Also the removal of objects from consideration needs to be changed in order to reflect the

partial coverage of objects. These changes are straightforward and therefore we omit a repeated listing of the modified algorithm here.

5 A k-Nearest Neighbor Classifier for Parallel Universes

Before we present our results, we shall briefly discuss an extension to the k-nearest neighbor approach for parallel universes, which we will use to compare our results against. It is based on the idea presented in [7], who use a modified distance measure to perform 3D object retrievals. Similar to learning in parallel universes they are given data sets in multiple descriptor spaces. The distance between a query object and an object in the training set is composed of their distances in the different universes. The individual distance values are weighted by the κ-*entropy impurity*, which is the class entropy of the κ nearest neighbors in the (labeled) training set in a universe around the query object, i.e. the object to be classified. The entropy impurity will be 0 if all κ neighbors in the training set have an equal class attribute and accordingly larger if the neighborhood contains objects of different classes. If $imp^{(u)}(q,\kappa)$ denotes the κ-entropy impurity in universe u for a query object q, then the accumulated overall distance $\delta(q,i)$ between q and and object i is [7]:

$$\delta(q,i) = \sum_{u=1}^{U} \frac{1}{1+imp^{(u)}(q,\kappa)} \cdot \frac{d^{(u)}(x_{q,u},x_{i,u})}{d^{(u)}_{\max_q}}.$$

Note the term $d^{(u)}_{\max_q}$ in the denominator of the distance coefficient: it is the maximum distance between q and the objects in the training set. It is used to overcome normalization problems that arise when accumulating distances from different domains/universes – a problem that the neighborgram approach fortunately does not suffer from as it does not compare distances across universes.

We use the above distance function in a k-nearest neighbor algorithm in the following section as it has proven to be appropriate for the 3D data set being analyzed [7]. However, its general applicability for parallel universe problems is questionable for mainly three reasons: (1) it does not scale for larger problem sizes (specifically because of query-based nearest neighbor searches in all universes), (2) it has the above-mentioned normalization problem (the normalization factor depends heavily on the data set), and (3) it does not produce an interpretable model in the form of rules or labeled clusters (as in the Neighborgram approach).

6 Results

We use a data set of 3D objects to demonstrate the practical usefulness of the presented neighborgram approach. The data set contains descriptions of 3D objects given in different universes, which cover image-, volume- and shape based properties [1,7]. There are a total of 292 objects, which were manually classified into 17 different classes such as airplanes, humans, swords, etc. The objects are described by means of 16 different universes, whose dimensions vary from 31 to more than 500 dimensions. The number of

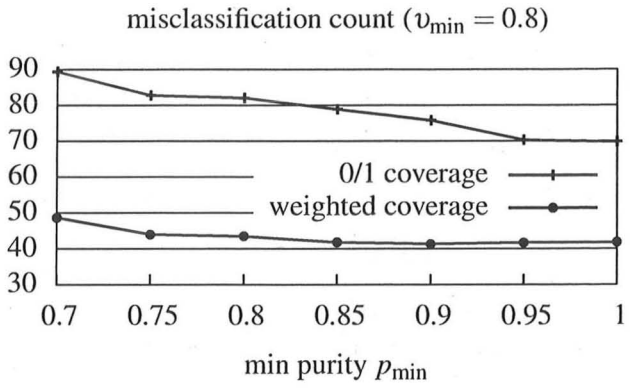

Fig. 2. Misclassification counts for sharp (0/1 coverage) vs. soft cluster boundaries (weighted coverage)

objects per class ranges from 9 to 56. The descriptors mainly fall into three categories. The image-based descriptors extract properties from an object's projections (typically after normalizing and aligning the object). They typically encode information regarding an object's silhouette or depth information. There are 6 universes in this category. Volume-based descriptors reflect volumetric properties of an object, for instance a distribution of voxels (small volume elements) in a unit cube being occupied by the object. There are 5 universes of this type. Finally, shape-based descriptors encode surface and curvature properties, e.g. based on the center points of the objects faces (center of the different polygons describing an object). The remaining 5 universes are of this type.

Similar to the results presented in [7] we use the Manhattan norm on the unnormalized attributes in each universe and perform 2-fold cross validation to determine error rates (we list absolute error counts for the 292 objects to be classified below).

We first ran a k-nearest neighbor approach to determine a reference value for the achievable error rate. We used the aggregated distance measure presented in section 5 and tested different settings. The smallest error we could achieve was 40 misclassifications for $\kappa = 3$ (to calculate a universe's κ-entropy impurity) and $k = 2$ (nearest neighbor parameter), which matches the findings of [7].

In a first experiment with the neighborgram clustering approach we tested the effect of the weighted coverage approach and compared it to the error rates when using a sharp cluster representation. We varied the minimum purity parameter p_{min} from 0.7 to 1.0 and set an overlap threshold $\upsilon_{min} = 0.8$. A minimum size r_{min} for a cluster was not set to respect underrepresented classes. Figure 2 shows the results. The 0/1 coverage curve indicates the error rates when using a sharp cluster representation, i.e. even objects at the cluster boundaries are fully covered. The weighted coverage approach improves the prediction quality considerably and yields error rates comparable to the k-nearest neighbor method. Note, using the weighted covering approach also increases the total number of clusters in the final model. If there are no further constraints regarding termination criterion or minimum cluster size, the cluster count can reach up to 600 clusters (from a total of $16 \cdot 292 = 4\,672$ cluster candidates).

In another experiment we evaluated the impact when identifying overlapping clusters across universes and adding those to the cluster set. This experiment compares the

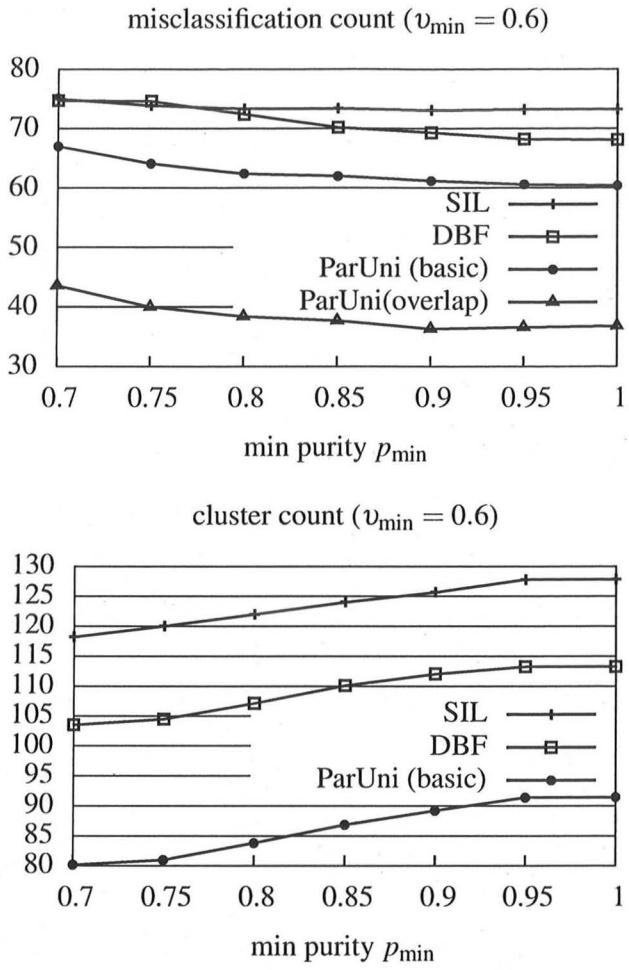

Fig. 3. Misclassification rates and cluster counts using the Parallel Universe extension compared to models built on individual universes

basic algorithm shown in algorithm 1 with the extension discussed in section 4.2 (using weighted coverage in either case). The results are summarized in figure 3 along with the results of the two best single-universe classifiers. These were built in the universes DBF (depth buffer) and SIL (silhouette), both of which are image-based descriptors. The basic algorithm in parallel universes already outperforms the single-universe classifiers in terms of both classification accuracy and numbers of clusters used. When additionally taking overlapping clusters into account (represented by the curve labeled "Paruni (overlap)"), the error rate drops considerably, showing the advantage of having some redundancy across universes in the cluster set. However, these re-occurring structures come at a price of an increased number of cluster, which is in the order of 500–600 (omitted in the graph). This suggests that the basic algorithm is suitable if the focus is on building an understandable model with possibly only a few clusters, whereas the enhanced algorithm with overlap detection may be appropriate when the focus lies on building a classifier with good prediction performance.

Neighborgram Visualization

Another advantage of the neighborgram data structure is the ability to visualize them and thus allow the user to visually assess a cluster's quality and potential overlaps across universes. Figure 4 shows an example. A row represents the neighborgrams for one object in four different universes (two image- (DBF & SIL), one shape- (SSD) and one volume-based (VOX)). We use the same visualization technique as in the illustrative example in figure 1, i.e. a distance plot, whereby the vertical stacking is only used to allow an individual selection of points. The plots show the 100 nearest neighbors; objects of the same class as the centroid are shown in dark grey and objects of all other classes in light grey. Objects of a (semi-automatically) selected cluster are additionally highlighted to see their occurrence also in the respective other neighborgrams/universes. 2D depictions of these objects are also shown at the bottom of the figures, whereby we manually expanded the cluster to also cover some conflicting objects (shown on the bottom right). Figure 4 shows the largest cluster in the DBF universe, which covers the class "swords". Note this cluster also groups in the SIL universe, which is also image-based, though there seems to be no grouping in the shape- and volume-based descriptors SSD and VOX. In contrast, clusters of the class "humans" form nicely in the shape-based universe SSD.

There are more interesting clusters in this data set, for instance larger groups of cars, which cluster well in the DBF and VOX universe or a group of weeds, which only clusters in the volume-based descriptor space. We do not show these clusters in separate figures due to space constraints. However, this demonstrates nicely that depending on the type of object certain descriptors, i.e. universes, are better suited to group them. These results emphasize that learning in parallel universes is an important and well suited concept when dealing with such types of information.

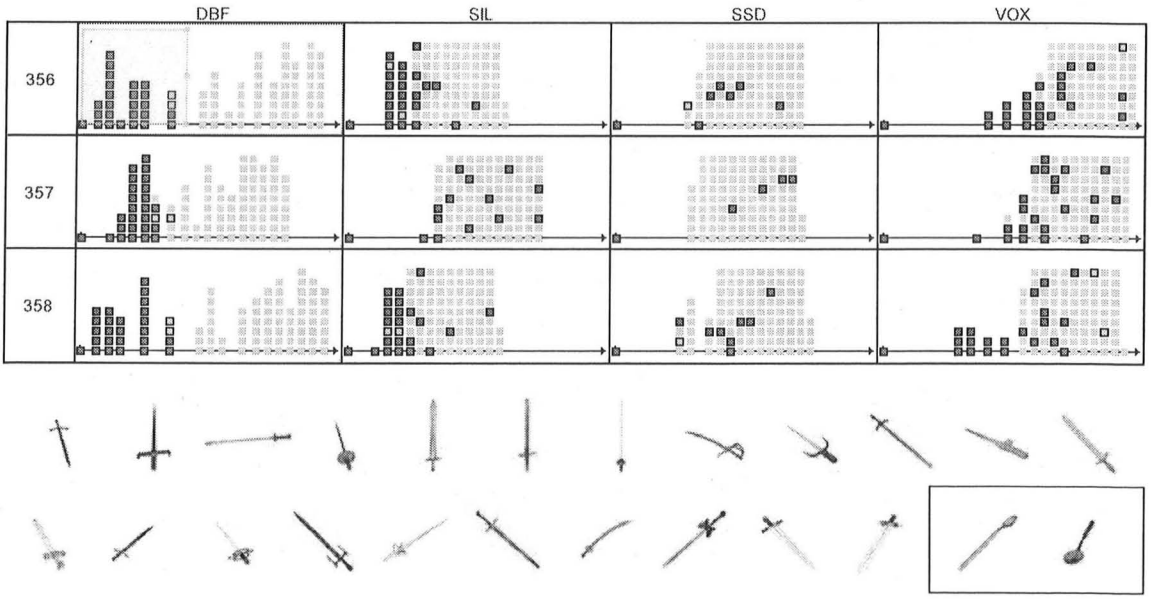

Fig. 4. The largest cluster in universe DBF covers objects of class "sword" (highlighted by border). 2D depictions of the covered objects are shown at the bottom. The cluster was manually expanded to cover also two conflict objects of class "spoon" (shown at bottom right).

7 Summary

We presented a supervised method for the learning in parallel universes, i.e. for the simultaneous analysis of multiple descriptor spaces using neighborgrams. We showed that by using an overlap criterion for clusters between universes we can considerably improve the prediction accuracy. Apart from using neighborgrams as an underlying basis to learn a classification model, they can also be used as visualization technique to either involve the user in the clustering process or to allow for a visual comparison of different universes.

References

1. Konstanz 3D model search engine (2008), `http://merkur01.inf.uni-konstanz.de/CCCC/` (last access March 20, 2009)
2. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proc. ACM SIGMOD International Conference on Management of Data, pp. 94–105. ACM Press, New York (1998)
3. Bender, A., Glen, R.C.: Molecular similarity: a key technique in molecular informatics. Organic and Biomolecular Chemistry 2(22), 3204–3218 (2004)
4. Berthold, M.R., Wiswedel, B., Patterson, D.E.: Neighborgram clustering: Interactive exploration of cluster neighborhoods. In: IEEE Data Mining, pp. 581–584. IEEE Press, Los Alamitos (2002)
5. Berthold, M.R., Wiswedel, B., Patterson, D.E.: Interactive exploration of fuzzy clusters using neighborgrams. Fuzzy Sets and Systems 149(1), 21–37 (2005)
6. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT 1998), pp. 92–100. ACM Press, New York (1998)
7. Bustos, B., Keim, D.A., Saupe, D., Schreck, T., Vranic, D.V.: Using entropy impurity for improved 3D object similarity search. In: Proceedings of IEEE International Conference on Multimedia and Expo (ICME 2004), pp. 1303–1306 (2004)
8. Bustos, B., Keim, D.A., Saupe, D., Schreck, T., Vranić, D.V.: An experimental effectiveness comparison of methods for 3D similarity search. International Journal on Digital Libraries, Special issue on Multimedia Contents and Management in Digital Libraries 6(1), 39–54 (2006)
9. Dasgupta, S., Littman, M.L., McAllester, D.A.: PAC generalization bounds for co-training. In: NIPS, pp. 375–382 (2001)
10. Friedman, J.H., Meulmany, J.J.: Clustering objects on subsets of attributes. Journal of the Royal Statistical Society 66(4) (2004)
11. Parsons, L., Haque, E., Liu, H.: Subspace clustering for high dimensional data: a review. SIGKDD Explor. Newsl. 6(1), 90–105 (2004)
12. Rüping, S., Scheffer, T. (eds.): Proceedings of the ICML 2005 Workshop on Learning With Multiple Views (2005)
13. Wiswedel, B., Berthold, M.R.: Fuzzy clustering in parallel universes. International Journal of Approximate Reasoning 45(3), 439–454 (2007)