

Algorithms and Applications for Spatial Data Mining

Martin Ester, Hans-Peter Kriegel, Jörg Sander (University of Munich)

1 Introduction

Due to the computerization and the advances in scientific data collection we are faced with a large and continuously growing amount of data which makes it impossible to interpret all this data manually. Therefore, the development of new techniques and tools that support the human in transforming data into useful knowledge has been the focus of the relatively new and interdisciplinary research area “knowledge discovery in databases”.

Knowledge discovery in databases (KDD) has been defined as the non-trivial process of discovering valid, novel, potentially useful and ultimately understandable patterns from data, a pattern is an expression in some language describing a subset of the data or a model applicable to that subset (Fayyad et al., 1996). The process of KDD is interactive and iterative, involving several steps such as data selection, data reduction, data mining, and the evaluation of the data mining results. The heart of the process, however, is the data mining step which consists of the application of data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data (Fayyad et al., 1996).

While a lot of research has been conducted on knowledge discovery and data mining in relational databases (see e.g. (Chen et al., 1996) or (Fayyad, 1997) for an overview), only a few works deal with knowledge discovery in *spatial* databases (see (Gueting, 1994) for an introduction to spatial databases, (Koperski et al., 1996) for an overview of spatial data mining). Finding implicit regularities, rules or patterns hidden in spatial databases is an important task, e.g. for geo-marketing, traffic control or environmental studies.

A spatial database contains objects which are characterized by a spatial location and/or extension as well as by several non-spatial attributes. Figure 7.1 illustrates a spatial database on Bavaria as an example. Depicted is the relation *Communities* containing polygons which represent communities in a geographic information system. This spatial database on Bavaria - referred to as the BAVARIA database - is used in some of the following sections as test database for our algorithms. The database contains the ATKIS 500 data and the Bavarian part of the statistical data obtained by the

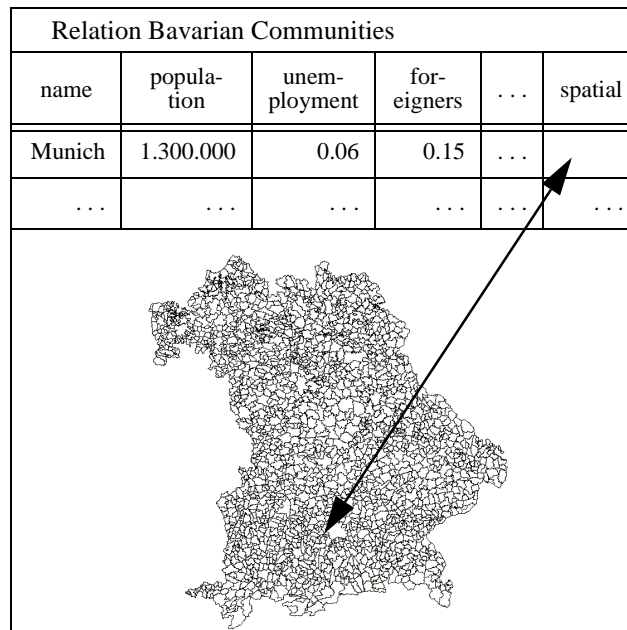


Figure 7.1. Spatial and non-spatial attributes of communities

German census of 1987, i.e. 2043 Bavarian communities with one spatial attribute (polygon) and 52 non-spatial attributes (such as average rent or rate of unemployment). Also included (in a separate table of the database) are spatial objects representing natural objects like mountains or rivers and infrastructure such as highways or railroads. The total number of spatial objects in the database then amounts to 6924.

The discovery process for spatial data is more complex than for relational data. This applies to both the efficiency of algorithms as well to the complexity of possible patterns that can be found in a spatial database. The reason is that, in contrast to mining in relational databases, spatial data mining algorithms have to consider the neighbours of objects in order to extract useful knowledge. This is necessary because the attributes of the neighbours of some object of interest may have a significant influence on the object itself.

In this chapter, we will first introduce, in section 7.2, a general framework for spatial data mining which takes into account the mentioned characteristics of spatial data. We also show that this approach allows a tight and efficient integration of spatial data mining algorithms with spatial database systems. In section 3 through 6, we then present algorithms for the tasks of spatial clustering, spatial characterization, spatial trend detection and spatial classification utilizing the proposed

framework. Furthermore, example applications are discussed for these algorithms. The last section gives a short summary and shows some directions for future research.

2 A Database-Oriented Framework for Spatial Data Mining

Our framework for spatial data mining is based on spatial neighbourhood relations between objects and on the induced neighbourhood graphs and neighbourhood paths which can be defined with respect to these neighbourhood relations. Thus, we introduce a set of database primitives or basic operations for spatial data mining which are sufficient to express most of the spatial data mining algorithms from the literature. This approach has several advantages. Similar to the relational standard language SQL, the use of standard primitives will speed-up the development of new data mining algorithms and will also make them more portable. Second, we can develop techniques to efficiently support the proposed database primitives (e.g. by specialized index structures) thus speeding-up all data mining algorithms which are based on our database primitives. Moreover, our basic operations for spatial data mining can be integrated into commercial database management systems. This will offer additional benefits for data mining applications such as efficient storage management, prevention of inconsistencies, index structures to support different types of database queries which may be part of the data mining algorithms.

2.1 Spatial Neighbourhood Relations, Spatial Neighbourhood Graphs and their Operations

Our database primitives for spatial data mining are based on the concepts of neighbourhood graphs and neighbourhood paths which in turn are defined with respect to neighbourhood relations between objects (Ester et al., 1997).

There are three basic types of spatial relations: topological, distance and direction relations which may be combined by logical operators to express a more complex neighbourhood relation. Spatial objects such as points, lines, polygons or polyhedrons are all represented by a set of points. For example, a polygon can be represented by its edges (vector representation) or by the points contained in its interior, e.g. the pixels of an object in a raster image (raster representation).

Topological relations are based on the boundaries, interiors and complements of the two related objects and are invariant under transformations which are continuous, one-one, onto and whose inverse is continuous. The relations are: *A disjoint B*, *A meets B*, *A overlaps B*, *A equals B*, *A covers*

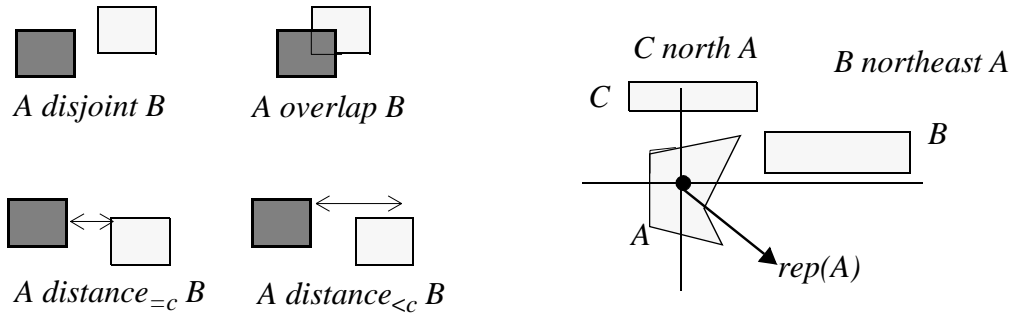


Figure 7.2. Illustration of some topological, some distance and some direction relations

B , A covered-by B , A contains B , A inside B . A formal definition has been given by Egenhofer (1991).

Distance relations compare the distance of two objects with a given constant using one of the arithmetic comparison operators. If $dist$ is a distance function, σ is one of the arithmetic predicates $<$, $>$ or $=$, and c is a real number, then a distance relation $O_1 \text{ distance}_{\sigma c} O_2$ between the two spatial objects O_1 and O_2 holds if $distance(O_1, O_2) \sigma c$.

To define the *direction relations*, e.g. $O_2 \text{ south } O_1$, we consider one representative point of the object O_1 as the origin of a virtual coordinate system whose quadrants and half-planes define the directions. To fulfil the direction predicate, *all points* of O_2 have to be located in the respective area of the plane. Figure 7.2 illustrates the definition of some direction relations using 2D polygons.

Obviously, the directions are not uniquely defined but there is always a smallest direction relation for two objects A and B , called the *exact direction relation* of A and B , which is uniquely determined. In figure 7.2, for instance, A and B satisfy the direction relations *northeast* and *east* but the exact direction relation of A and B is *northeast*.

By combining basic spatial relations via logical operators it is possible to define more complex spatial relations, e.g. “ O_1 being north of O_2 and no more than 5 km away”. Each such spatial relation induces a spatial neighbourhood graph as defined in the following definition.

Definition 1: (neighbourhood graphs and paths)

Let *neighbour* be a neighbourhood relation and *DB* be a database of spatial objects. A *neighbourhood graph* $G_{neighbour}^{DB} = (N, E)$ is a graph with nodes $N = DB$ and edges $E \subseteq N \times N$ where an edge $e = (n_1, n_2)$ exists iff *neighbour*(n_1, n_2) holds. A *neighbourhood path* of length k is defined as a sequence of nodes $[n_1, n_2, \dots, n_k]$, where *neighbour*(n_i, n_{i+1}) holds for all $n_i \in N, 1 \leq i < k$.

We assume the standard operations from relational algebra like *selection*, *union*, *intersection* and *difference* to be available for sets of objects and sets of neighbourhood paths (e.g., the operation *selection*(*set*, *predicate*) returns the set of all elements of a *set* satisfying the predicate *predicate*). In addition, we introduce some operations which are specific to neighbourhood graphs and paths and which are designed to support spatial data mining. The following operations are briefly described:

- *neighbours*: $\text{Graphs} \times \text{Objects} \times \text{Predicates} \rightarrow \text{Sets_of_objects}$
- *paths*: $\text{Sets_of_objects} \rightarrow \text{Sets_of_paths}$;
- *extensions*: $\text{Graphs} \times \text{Sets_of_paths} \times \text{Integer} \times \text{Predicates} \rightarrow \text{Sets_of_paths}$

The operation *neighbours*(*graph*, *object*, *predicate*) returns the set of all objects connected to *object* in *graph* satisfying the conditions expressed by the predicate *predicate*.

The operation *paths*(*objects*) creates all paths of length 1 formed by a single element of *objects* and the operation *extensions*(*graph*, *paths*, *length*, *predicate*) returns the set of all paths of the specified *length* in *graph* extending one of the elements of *paths*. The extended paths must satisfy the predicate *predicate*. The elements of *paths* are not contained in the result implying that an empty result indicates that none of the elements of *paths* could be extended.

Because the number of neighbourhood paths may become very large, the argument *predicate* in the operations *neighbours* and *extensions* acts as a filter to restrict the number of neighbours and paths to certain types of neighbours or paths. The definition of *predicate* may use spatial as well as non-spatial attributes of the objects or paths.

For the purpose of KDD, we are mostly interested in paths “leading away” from the start object. We conjecture that a spatial KDD algorithm using a set of paths which are crossing the space in arbitrary ways will not produce useful patterns. The reason is that spatial patterns are most often the effect of some kind of influence of an object on other objects in its neighbourhood. Furthermore, this influence typically decreases or increases more or less continuously with increasing or

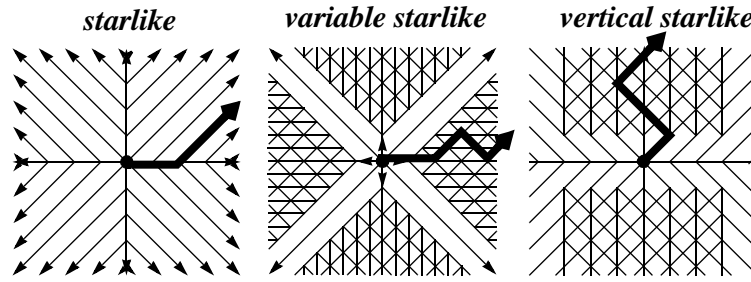


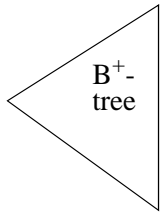
Figure 7.3. Illustration of some filter predicates

decreasing distance. To create only “relevant” paths, we introduce special filter predicates which select only particular subsets of all paths, i.e. paths which are “*leading away*” from the start objects in a certain sense. This approach also significantly reduces the runtime of the spatial data mining algorithms operating on neighbourhood paths.

There are different possibilities to define filters for paths “*leading away*” from start objects. The filter *starlike*, e.g., is a very restrictive filter which allows only a small number of “coarse” paths. This filter is appropriate for many applications, and in terms of runtime it is most efficient. It requires that, when extending a path $p = [n_1, n_2, \dots, n_k]$ with a node n_{k+1} , the exact “final” direction of p may not be generalized. For instance, a path with final direction *northeast* can only be extended by a node of an edge with the exact direction *northeast*. The filter *variable-starlike* allows more “fine-grained” paths by requiring only that, when extending p the edge (n_k, n_{k+1}) has to fulfil *at least* the exact “initial” direction of p . For instance, a neighbourhood path with initial direction *north* can be extended such that the direction *north* or the more special direction *northeast* is satisfied. The filter *variable-starlike* allows a more detailed spatial analysis than the filter *starlike*, but it increases the runtime of a data mining algorithm because more paths have to be processed by the algorithm. Figure 7.3 illustrates these filters when extending the paths from a given start object. The figure also depicts another filter *vertical starlike* which is less restrictive in vertical than in horizontal direction. This filter is appropriate when the vertical direction should be analyzed in greater detail than the horizontal direction.

2.2 Efficient DBMS Support

In this section, we present techniques to efficiently support the proposed database primitives by a DBMS. Thus, our basic operations for spatial data mining can be integrated into commercial DBMSs and all the benefits of these systems can be exploited for spatial data mining applications.



Object-ID	Neighbour	Distance	Direction	Topology
o_1	o_2	2.7	southwest	disjoint
o_1	o_3	0	northwest	overlap
...

Figure 7.4. Sample Neighbourhood Index

Typically, spatial index structures, e.g. R-trees (Guttman, 1984), are used in an DBMS to speed up the processing of spatial queries such as region queries or nearest neighbour queries (Gueting, 1994). If the spatial objects are fairly complex, however, retrieving the neighbours of some object this way is still very time consuming due to the complexity of the evaluation of neighbourhood relations on such objects. Furthermore, when creating all neighbourhood paths with a given source object, a very large number of `neighbours` operations have to be performed. Finally, many spatial databases are rather static since there are not many updates on objects such as geographic maps or proteins. Therefore, materializing the relevant neighbourhood graphs and avoiding access to the spatial objects themselves may be worthwhile. Our concept of *neighbourhood indices* is related to that of the distance associated join indices (Lu and Han, 1992) with the following new contributions:

- A specified maximum distance restricts the pairs of objects represented in a neighbourhood index.
- For each of the different types of neighbourhood relations (that is distance, direction, and topological relations), the concrete relation of the pair of objects is stored.

In the following, we introduce neighbourhood indices more formally.

Definition 2: (*neighbourhood index*)

Let DB be a set of spatial objects and let max and $dist$ be real numbers. Let D be a direction relation and T be a topological relation. Then the *neighbourhood index* for DB with maximum distance max , denoted by I_{max}^{DB} , is defined as follows:

$$I_{max}^{DB} = \{(O_1, O_2, dist, D, T) \mid O_1, O_2 \in DB, O_1 \text{ distance}_{=dist} O_2 \wedge dist \leq max \wedge O_2 D O_1 \wedge O_1 T O_2\}.$$

A simple implementation of a neighbourhood index using a B⁺-tree on the key attribute *Object-ID* is illustrated in figure 7.4.

A neighbourhood index supports not only one but a set of neighbourhood graphs. We call a neighbourhood index *applicable* for a given neighbourhood graph if the index contains an entry for each of the edges of the graph. To find the neighbourhood indices applicable for some neighbourhood graph, we introduce the notion of the critical distance of a neighbourhood relation r . Intuitively, the *critical distance* of a neighbourhood relation r is the maximum possible distance for a pair of objects O_1 and O_2 satisfying $O_1 r O_2$. The following lemma allows us to calculate the critical distance for any neighbourhood relation. The critical distance is calculated recursively along the composition of a neighbourhood relation.

Lemma 1: The following equation holds for the critical distance of a neighbourhood relation r :

$$c\text{-distance}(r) = \begin{cases} 0 & \text{if } r \text{ is a topological relation except } disjoint \\ d & \text{if } r \text{ is the relation } distance_{<d} \text{ or } distance_{=d} \\ \infty & \text{if } r \text{ is a direction relation, the relation } distance_{>d} \\ & \text{or } disjoint \\ \min(c\text{-distance}(r_1), c\text{-distance}(r_2)) & \text{if } r = r_1 \wedge r_2 \\ \max(c\text{-distance}(r_1), c\text{-distance}(r_2)) & \text{if } r = r_1 \vee r_2 \end{cases}$$

A neighbourhood index with a maximum distance of max is applicable for a neighbourhood graph with relation r if the critical distance of r is not larger than max . Obviously, if two neighbourhood indices $I_{c_1}^{DB}$ and $I_{c_2}^{DB}$ with $c_1 < c_2$ are available and applicable, using $I_{c_1}^{DB}$ is more efficient because in general it has less entries than $I_{c_2}^{DB}$. The *smallest applicable neighbourhood index* for some neighbourhood graph is the applicable neighbourhood index with the smallest critical distance.

In figure 5, we sketch the algorithm for processing the *neighbours* operation which makes use of the smallest applicable neighbourhood index. The first step, the *index selection*, selects a neighbourhood index. If there is no applicable neighbourhood index, then the standard approach of using some spatial index structure is followed. The *filter step* returns a set of candidate objects (which may satisfy the specified neighbourhood relation) with a cardinality significantly smaller than the database size. In the last step, the *refinement step*, for all these candidates the neighbourhood relation as well as the additional predicate $pred$ are evaluated and all objects passing this test are returned as the resulting neighbours.

To implement the *extensions* operation, we perform a depth-first search. Thus, a path buffer of size $O(max\text{-length})$ is sufficient to store the intermediate results. On the other hand, a breadth-first

```

neighbours (graph  $G_r^{DB}$ , object  $o$ , predicate  $pred$ )
  select as  $I$  the smallest applicable neighbourhood index for  $G_r^{DB}$ ; // Index Selection
  if such  $I$  exists then // Filter Step
    use the neighbourhood index  $I$  to retrieve as candidates the set of objects  $c$ 
    having an entry  $(o, c, dist, D, T)$  in  $I$ 
  else use the spatial index for  $DB$  to retrieve as candidates
    the set of objects  $c$  satisfying  $o r c$ ;
  initialize an empty set of neighbours; // Refinement Step
  for each  $c$  in candidates do
    if  $o r c$  and  $pred(c)$  then
      add  $c$  to neighbours
  return neighbours;

```

Figure 7.5. Algorithm neighbours

search would require a much larger buffer size since it begins creating all paths before finishing the first one. To retrieve the nodes for potential extensions of a candidate path, the *neighbours* operation is used indicating that the efficiency of this operation is crucial.

To create a neighbourhood index I_{max}^{DB} , a spatial join on DB with respect to the neighbourhood relation $(O_1 distance_{=dist} O_2 \wedge dist \leq max)$ is performed. A spatial join can be efficiently processed by using a spatial index structure. For each pair of objects returned by the spatial join, we then have to determine the exact distance, the direction relation and the topological relation. The resulting tuples of the form $(O_1, O_2, Distance, Direction, Topology)$ are stored in a relation which is indexed by a B-tree on the attribute O_1 .

Updates of a database, i.e. insertions or deletions of spatial objects, require updates of the derived neighbourhood indices. Fortunately, the update of a neighbourhood index I_{max}^{DB} is restricted to the neighbourhood of the respective object defined by the neighbourhood relation $A distance_{<max} B$. This neighbourhood can be efficiently retrieved by using either a neighbourhood index (in case of a deletion) or by using a spatial index structure (in case of an insertion). That means, it is not required to re-build a neighbourhood index from scratch when an update occurs. Instead, entries can “incrementally” be inserted into or deleted from a neighbourhood index.

The database primitives were implemented on top of the commercial DBMS Illustra, Version 3.3, using its 2D spatial data blade which provides R-trees. Neighbourhood indices were realized in the DBMS by tables and B-trees as described above (see figure 7.4) and the geographic database

of Bavaria was used for an experimental performance evaluation. The results of our performance evaluation (Ester et al., 2000) demonstrate a significant speed-up for the *neighbours* operation with compared to without neighbourhood index. The *neighbours* operation is crucial to the efficient DBMS support of our database primitives since the implementation of the operations for extending neighbourhood paths is based on the *neighbours* operation.

In particular, a neighbourhood index is very efficient for complex spatial objects (i.e. large numbers of vertices per spatial object) and an average number of neighbours per spatial object which is not too large. This is a typical setting, e.g., for geographic information systems when using *intersects* as the neighbourhood predicate. If the average number of neighbours is very large for a spatial neighbourhood relation, we also have to consider the size of the neighbourhood index. In the worst case, depending on the critical distance $c\text{-distance}(r)$ and the distribution of the spatial objects, the storage requirements for the materialization of a neighbourhood index can be of $O(n^2)$. Ester et al. (2000) present a cost model for the neighbours operation. Based on system parameters (e.g. average execution time for a page access) and information about the neighborhood relation (e.g. average number of neighbours) the model can be used to predict the performance gain of a neighbourhood index and whether this time gain is worth the additional storage requirements.

3 Spatial Clustering

Clustering is the task of grouping the objects of a database into meaningful subclasses (that is, clusters) so that the members of a cluster are as similar as possible whereas the members of different clusters differ as much as possible from each other. Applications of clustering in spatial databases are, e.g., the detection of seismic faults by grouping the entries of an earthquake catalog or the creation of thematic maps in geographic information systems by clustering feature vectors. We can support clustering algorithms by our database primitives if the clustering algorithm is based on a “local” cluster condition, i.e. if it constructs clusters by analyzing a restricted neighbourhood of the objects. Examples are the density-based clustering algorithm DBSCAN (Ester et al., 1996) as well as its generalized version GDBSCAN (Sander et al., 1998) which is discussed in the following.

3.1 Generalized Density-Based Clustering

GDBSCAN (*Generalized Density Based Spatial Clustering of Applications with Noise*) relies on a density-based notion of clusters. The key idea of a density-based cluster is that for each point of a cluster its *Eps*-neighbourhood for some given $Eps > 0$ has to contain at least a minimum number of points, i.e. the “density” in the *Eps*-neighbourhood of points has to exceed some threshold.

“Density-based clusters” can be generalized to density-connected sets in the following way: First, any notion of a neighbourhood can be used instead of an *Eps*-neighbourhood if the definition of the neighbourhood is based on a binary predicate *NPred* which is symmetric and reflexive. Second, instead of simply counting the objects in a neighbourhood of an object, other measures to define an equivalent of the “cardinality” of that neighbourhood can be used as well. For that purpose we assume a predicate *MinWeight* which is defined for sets of objects and which is *true* for a neighbourhood if the neighbourhood has the minimum weight (e.g. a minimum cardinality as for density-based clusters).

Whereas a distance-based neighbourhood is a natural notion of a neighbourhood for point objects, it may be more appropriate to use topological relations such as *intersects* or *meets* to cluster spatially extended objects such as a set of polygons of largely differing sizes. There are also specializations equivalent to simple forms of region growing (Niemann, 1990), i.e. only local criteria for expanding a region can be defined by the weighted cardinality function. For instance, the neighbourhood may be given simply by the neighbouring cells in a grid and the weighted cardinality function may be some aggregation of the non-spatial attribute values. While region growing algorithms are highly specialized to pixels, density-connected sets can be defined for any data types. The neighbourhood predicate *NPred* and the *MinWeight* predicate for these specializations are listed below and illustrated in figure 7.6 (see (Sander et al., 1998) for a more detailed discussion of neighbourhood relations for different applications):

- Density-based clusters: *NPred*: “distance $\leq Eps$ ”, *wCard*: cardinality, *MinWeight*(*N*): $|N| \geq MinPts$
- Clustering of polygons: *NPred*: “*intersects*” or “*meets*”, *wCard*: sum of areas, *MinWeight*(*N*): sum of areas $\geq MinArea$

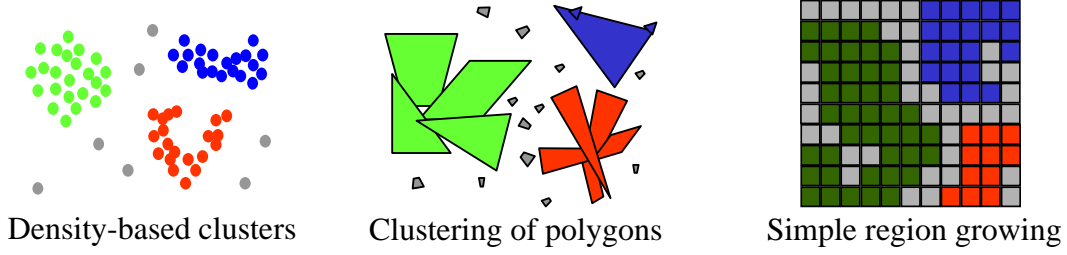


Figure 7.6. Different specializations of density-connected sets

GDBSCAN (DB, NPred, MinWeight)

```
// Precond.: for all objects  $o$  in DB:  $o.CId = UNCLASSIFIED$  and  $o.Processed = FALSE$ 
NPredGraph := createNeighbourhoodGraph(DB, NPred)
ClusterId := nextId(NOISE);
for  $i$  from 1 to DB.size do
  Object := DB.get( $i$ );
  if not Object.Processed then
    if ExpandCluster(Object, ClusterId, NPredGraph, MinWeight) then
      ClusterId := nextId(ClusterId)
```

Figure 7.7. Algorithm GDBSCAN

- Simple region growing: $NPred$: “neighbour and similar non-spatial attributes”,
 $MinWeight(N)$: $aggr(\text{non-spatial values}) \geq \text{threshold}$.

All instances of density-connected sets can be constructed by the same algorithm which is based on a neighbourhood graph G_{NPred}^{DB} and which uses the *neighbours* operation introduced above in our framework for spatial data mining. To find a density-connected set, GDBSCAN starts with an arbitrary object p and retrieves all objects density-reachable from p with respect to $NPred$ and $MinWeight$. Density-reachable objects are retrieved by performing successive $NPred$ -neighbourhood queries and checking the minimum weight of the respective results. If p is a core object, this procedure yields a density-connected set with respect to $NPred$ and $MinWeight$. If p is not a core object, no objects are density-reachable from p and p is assigned to *NOISE*. This procedure is iteratively applied to each object p which has not yet been classified. Thus, a density-based decomposition of a dataset is constructed.

In figure 7.7, we present the basic version of the algorithm GDBSCAN based on a neighbourhood graph with respect to the neighbourhood predicate $NPred$:

```

ExpandCluster(Object, CId,  $G_{NPred}^{DB}$ , MinWeight):Boolean
  neighbours := neighbours(  $G_{NPred}^{DB}$ , Object, true);
  Object.Processed := true;
  if MinWeight(neighbours) then // object is a core object
    Seeds.init(NPred, MinWeight, CId);
    Seeds.update(neighbours, Object);
    while not Seeds.empty() do
      currentObject := Seeds.next();
      neighbours := neighbours(  $G_{NPred}^{DB}$ , currentObject, true);
      currentObject.Processed := true;
      if MinWeight(neighbours) then
        Seeds.update(neighbours, currentObject);
    return true;
  else // Object is NOT a core object
    changeClusterId(Object, NOISE);
    return false;

```

Figure 7.8. Function ExpandCluster

SetOfObjects is either the whole database or a discovered cluster from a previous run. *NPred* and *MinWeight* are the global density parameters. The cluster identifiers are from an ordered and countable data-type (e.g. implemented by Integers) where *UNCLASSIFIED* < *NOISE* < “other Ids”, and each object will be marked with a cluster-id *Object.CId*. The function *nextId(clusterId)* returns the successor of *clusterId* in the ordering of the data-type (e.g. implemented as *Id := Id + 1*). The function *SetOfObjects.get(i)* returns the *i*-th element of *SetOfObjects*. In figure 7.8, function *ExpandCluster* constructing a density-connected set for a core object *Object* is presented.

A call of *neighbours(NPredGraph, Object, true)* returns the *NPred*-neighbourhood of *Object* by using the neighbourhood graph *NPredGraph*. If the *NPred*-neighbourhood of *Object* has minimum weight, the objects from this *NPred*-neighbourhood are inserted into the set *Seeds* and the function *ExpandCluster* successively performs *NPred*-neighbourhood queries for each object in *Seeds*, thus finding all objects that are density-reachable from *Object*, i.e. constructing the density-connected set that contains the core object *Object*.

The class *Seeds* controls the main loop of *ExpandCluster*. The method *Seeds.next()* selects the next element from the set *Seeds* and deletes it from *Seeds*. The method *Seeds.update(neighbors, centreObject)* inserts into the class *Seeds* all objects from the set *neighbors* which have not yet been considered, i.e. which have not already been found to belong to the current density-connected

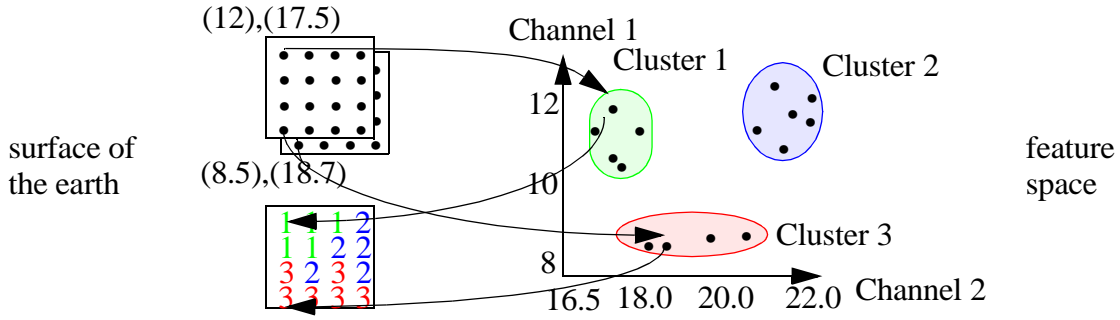


Figure 7.9. Relation between 2D image and feature space

set. This method also calls the method to change the cluster-id of the objects to the current clusterId.

3.2 Applications

Application 1: Earth Science (5D points)

In this application, we use a 5-dimensional feature space obtained from several satellite images of a region on the surface of the earth covering California. These images are taken from the raster data of the SEQUOIA 2000 Storage Benchmark. After some preprocessing, five images containing 1,024,000 intensity values (8 bit pixels) for 5 different spectral channels for the same region were combined. Each pixel corresponds to an earth surface area of 1,000 by 1,000 meters and a 5-dimensional feature vector is assigned to this area.

Finding clusters in such feature spaces is a common task in remote sensing digital image analysis (Richards, 1983) for the creation of thematic maps in geographic information systems. The assumption is that feature vectors for areas of the same type of underground on the earth are forming groups in the high dimensional feature space (see figure 7.9 illustrating the case of 2D raster images).

We used “ $\text{dist}(X,Y) < 1.42$ ” as $NPred(X,Y)$ and “ $\text{cardinality}(N) \leq 20$ ” as $MinWeight(N)$. After some postprocessing we obtained 9 clusters with sizes ranging from 598,863 to 2,016 points. The postprocessing consisted of two steps: 1. rejecting clusters containing less than 200 points and 2. re-assigning the points from the rejected clusters and the noise points to the remaining clusters because a non-noise class label for each raster point is required for this application.

The result is shown in figure 7.10. Each cluster was coded by a different color. Then each point in the image of the surface of the earth was colored according to the identifier of the cluster containing the corresponding 5-dimensional vector. A high degree of correspondence between the obtained image and a physical map of California can easily be seen. A detailed discussion of this correspondence is beyond the scope of this chapter.

This is a simplified description of the application. In practice, to obtain high quality results, there are several pre- and post-processing steps to the clustering of the feature vectors. Especially, the clustering of the feature vectors should be followed by a spatial smoothing because feature vectors which are close to each other in space are also likely to belong to the same class.

Application 2: Geography (2D polygons)

In the following, we present a simple method for detecting “influence regions” in a geographic database. GDBSCAN is used to extract density-connected sets of neighbouring objects having a similar value of the non-spatial attribute(s). To define the similarity on an attribute, its domain is partitioned into a number of disjoint classes and values in the same class are considered similar to each other. The sets with the highest or lowest attribute value(s) are most interesting and are called *influence regions*, i.e. the maximal neighbourhood of a centre having a similar value in the non-spatial attribute(s) as the centre itself. For economic geography, the resulting influence region may be further analyzed by comparing them to a circular region representing the expected theoretical shape to obtain a possible deviation. Different methods may be used for this comparison. A *difference-based method* calculates the difference of both, the observed influence region and the theoretical circular region, thus returning some region indicating the location of a possible deviation. An *approxima-*

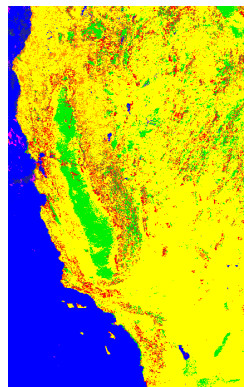


Figure 7.10. Visualization of the clustering result for the SEQUIOA 2000 raster data

tion-based method calculates the optimal approximating ellipsoid of the observed influence region. If the two main axes of the ellipsoid differ in length significantly, then the longer one is returned indicating the direction of a deviation. Ester et al. (1997) present a detailed description of this application.

GDBSCAN can be used to extract the influence regions from a spatial database. We define $NPred(X,Y)$ as “ $\text{intersect}(X,Y) \wedge \text{attr-class}(X) = \text{attr-class}(Y)$ ” and use cardinality as $wCard$ function. Furthermore, we set $MinCard$ to 2 in order to exclude sets of less than 2 objects. Some results of this approach for the BAVARIA database are illustrated in figure 7.11 .

4 Spatial Characterization

The task of *characterization* is to find a compact description for a selected subset of the database. In this section, we discuss the task of characterization in the context of spatial databases and present two relevant methods.

4.1 Algorithms

The task of mining association rules has been introduced by Agrawal and Srikant (1994). An *association rule* is a rule $I_1 \Rightarrow I_2$ where I_1 and I_2 are disjoint sets of items. The *support* of the rule is given by the number of database tuples containing all elements of I_1 and the *confidence* is given by the number of tuples containing all elements of both I_1 and I_2 . For a database DB of transactions (i.e. records contain sets of items bought by some customer in one transaction), all association rules should be discovered having a support of at least *minsupp* and a confidence of at least *minconf* in DB .

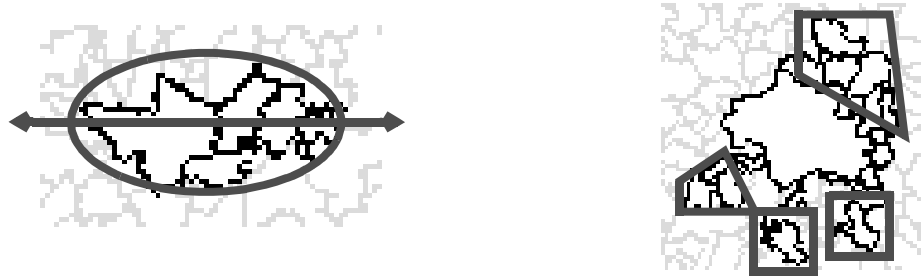


Figure 7.11. Illustration of the influence regions of Ingolstadt (left) and Munich (right) and their deviation from the expected shape

Extending the general concept of association rules, Koperski and Han (1995) introduce *spatial association rules* which describe associations between objects based on spatial neighbourhood relations. For instance, a user may want to discover the spatial associations of towns in British Columbia with roads, waters or boundaries having some specified support and confidence. Figure 7.12 depicts the specification of this spatial data mining task.

discover spatial association rules
inside British_Columbia
from road R, water W, mines M, boundary B
in relevance to town T
where close-to(T.geo, X.geo) **and** X in {R, W, M, B}
having minsupp = 5 % **and** minconf = 80 %

Figure 7.12. Example specification for mining spatial association rules

Then, the following spatial association rule may be discovered:

$\forall X \in DB \exists Y \in DB: \text{is-a}(X, \text{town}) \rightarrow \text{close-to}(X, Y) \wedge \text{is-a}(Y, \text{water}) \text{ (80\%)}$

This rule states that 80% of the selected towns are close to water, i.e. the rule characterizes towns in British Columbia as generally being close to some lake, river etc.

The input for mining spatial association rules specifies a relation of n tuples with a spatial attribute, a spatial neighbourhood relation, a concept hierarchy for each of the attributes, a selection of relevant object types, the minimum support, and the minimum confidence.

The proposed algorithm consists of five steps. Step 2 (coarse spatial computation) and step 4 (refined spatial computation) involve spatial aspects of the objects and thus are examined in the following. Step 2 computes spatial joins of the object type to be characterized (such as town) with each of the other specified object types (such as water, road, boundary or mine) using the neighbourhood relation (such as close-to). For each of the candidates obtained from step 2 which passed step 3, in step 4 the exact spatial relation, for example overlap, is determined. Finally, a relation such as the one depicted in figure 7.13 results which is the input for the final, non-spatial, Apriori-like step of rule generation. To implement this algorithm using our database primitives it is sufficient to replace step 2 by the following procedure. The spatial join can be replaced by calling a *neighbours* operation for each target object selected in step 1. The under-

lying neighbourhood graph in this case is defined by the user-specified neighbourhood relation (e.g. close-to).

Town	Water	Road	Boundary
Saanich	<meet, J.FucaStrait>	<overlap,highway1>, <close-to,highway17>	<close-to,US>
PrinceGeorge		<overlap, highway97>	
Petincton	<meet,OkanaganLake>	<overlap, highway97>	<close-to,US>
...

Figure 7.13. Input for the step of rule generation

As shown by Koperski and Han (1995), the proposed algorithm can be efficiently implemented, and it is included in the *GeoMiner* system (Han et al., 1997). A limitation of spatial association rules is that they do not use the non-spatial attributes to characterize the specified objects.

We define a *spatial characterization* of a set of target objects with respect to the database containing them as a description of the spatial and non-spatial properties which are typical for the target objects but not for the whole database. We use the relative frequencies of the non-spatial attribute values and the relative frequencies of the different object types as the interesting properties.

To obtain a *spatial* characterization, we consider not only the properties of the target objects, but also the properties of their neighbours up to a given maximum number of edges in the neighbourhood graph. Figure 7.14 illustrates relative frequencies in the database as well as in the target regions and the ratio of these frequencies in comparison with the specified level of significance.

We define a *property* of some spatial object by a tuple of one of the two following formats: either (*attribute*, *value*) or (“type”, type) where “type” denotes the special “attribute” representing the object type such as community, mountain or river. The task of spatial characterization is to discover the set of all properties for which the relative frequency in a set *targets*, extended by its neighbours, is significantly different from the relative frequency in *DB*. A very frequent property present only in the neighbourhood of very few of the *targets* would create misleading results.

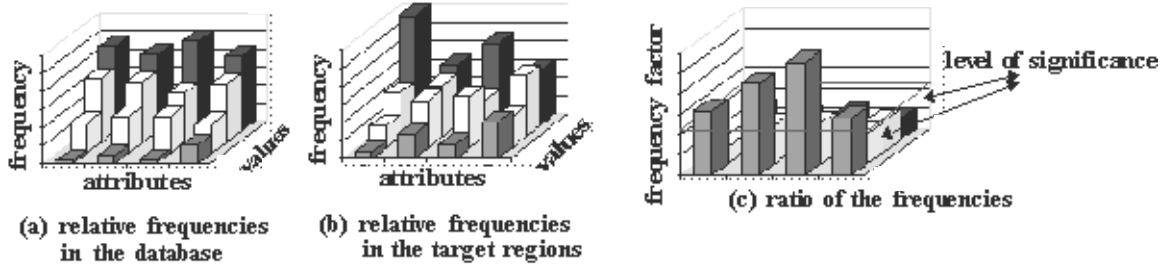


Figure 7.14. Sample frequencies and ratios

Therefore, we also require that such a property must have a significantly different, either smaller or larger, relative frequency in the neighbourhood of at least *proportion* many targets.

Definition 3: (spatial characterization): Let $G_{neighbor}^{DB}$ be a neighbourhood graph and *targets* be a subset of *DB*. Let $freq^s(prop)$ denote the number of occurrences of the property *prop* in the set *s* and let $card(s)$ denote the cardinality of *s*. The *frequency factor* of *prop* with respect to *targets* and *DB*, denoted by $f_{targets}^{DB}(prop)$, is defined as follows:

$$f_{targets}^{DB}(prop) = \frac{freq^{targets}(prop)}{card(targets)} / \frac{freq^{DB}(prop)}{card(DB)}$$

Let *significance* and *proportion* be real numbers and let *max-neighbours* be a natural number. Let $neighbors_G^i(s)$ denote the set of all objects reachable from one of the elements of *s* by traversing at most *i* of the edges of the neighbourhood graph *G*. Then, the task of *spatial characterization* is to discover each property *prop* and each natural number $n \leq max-neighbours$ such that (1) the set $objects = neighbors_G^n(targets)$ as well as (2) the sets $objects = neighbors_G^n(\{t\})$ for at least *proportion* many $t \in targets$ satisfy the condition:

$$f_{objects}^{DB}(prop) \geq significance \quad \text{or} \quad f_{objects}^{DB} \leq \frac{1}{significance}$$

In point (1) the union of the neighbourhood of all target objects is considered simultaneously, whereas in point (2) the neighbourhood of each targets is considered separately. The parameter *proportion* specifies the minimum confidence required for the characterization rules and the frequency factors of the properties provide a measure of their interestingness with respect to the target objects.

Figure 7.15 presents the algorithm for discovering spatial characterizations. The parameter *proportion* is relevant only for the last step of the algorithm, i.e. for the generation of a rule. Note the

```

discover-spatial-characterization(graph  $G_r^{DB}$  ; set of objects targets;  

                                     real significance, proportion; integer max-neighbours)  

initialize the set of characterizations as empty;  

initialize the set of regions to targets;  

initialize n to 0;  

calculate  $frequency^{DB}(prop)$  for all properties  $prop = (attribute, value)$  in DB;  

while  $n \leq max-neighbours$  do  

    for each attribute of DB and for the special attribute object type do  

        for each value of attribute do  

            calculate  $frequency^{regions}(prop)$  for property  $prop = (attribute, value)$ ;  

            if  $f_{regions}^{DB}(prop) \geq significance$  or  $f_{regions}^{DB}(prop) \leq 1 / significance$  then  

                add  $(prop, n, f_{regions}^{DB}(prop))$  to the set characterizations;  

            if  $n < max-neighbours$  then  

                for each object in regions do  

                    add  $neighbours(G_r^{DB}, object, true)$  to regions;  

                increment n by 1;  

extract all tuples  $(prop, n, factor)$  from characterizations which are significant in at  

least proportion of the regions with n extensions;  

return the rule generated from these characterizations;

```

Figure 7.15. Algorithm discover-spatial-characterization

importance of the parameter *max-neighbours* (that is, the maximum number of edges of the neighbourhood graph traversed starting from a target object) in the resulting characterizations. For example, a property may be significant when considering all neighbours which are reachable from one of the target objects via 2 edges of the neighbourhood graph. However, the same property may not be significant when considering further neighbours if then the target regions are extended by objects for which the property is not frequent. The generated rule has the following format:

$$target \Rightarrow p_1(n_1, freq-fac_1) \wedge \dots \wedge p_k(n_k, freq-fac_k).$$

This rule means that for the set of all targets extended by n_i neighbours, the property p_i is $freq-fac_i$ times more (or less) frequent than in the database.

4.2 Application: Economic Geography

A geographic database may be used, e.g., by economic geographers to discover spatial characterizations of the economic power or other interesting properties of communities. Some non-spatial attribute such as the unemployment rate is chosen as a relevant indicator.

The BAVARIA database was used for an experimental performance evaluation of the algorithms (Ester et al., 1998).

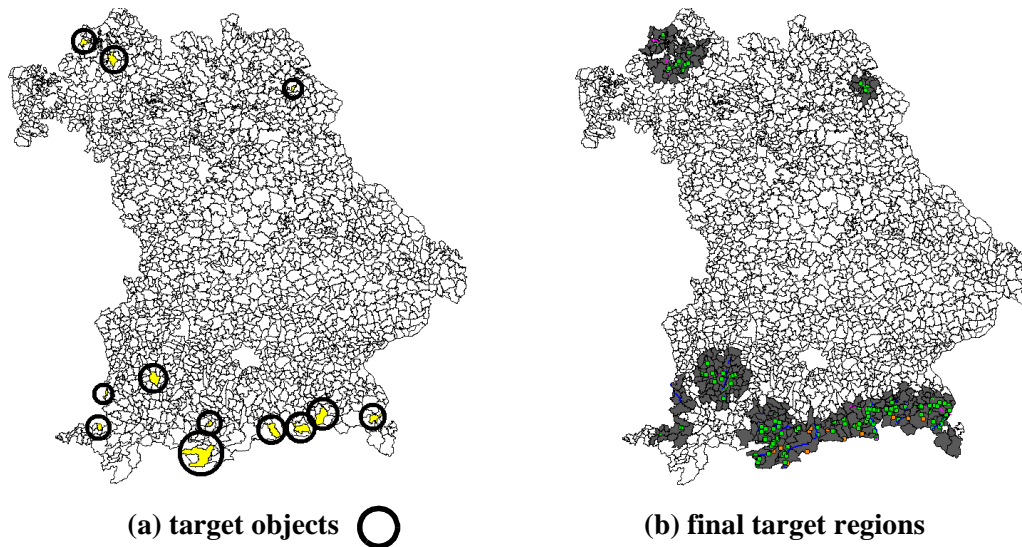


Figure 7.16. Characterization wrt. high proportion of retired persons

The characterization algorithm usually starts with a small set of target objects, selected for instance by a condition on some non-spatial attribute(s) such as “proportion of retired persons = HIGH” (see figure 7.16 (a)). Then, the algorithm expands regions around the target objects, simultaneously selecting those attributes of the regions for which the distribution of values differs significantly from the distribution in the whole database (figure 7.16 (b)).

In the last step of the algorithm, a characterization rule is generated describing the target regions (figure 7.17). Informally, this rule states that “retirees prefer somewhat rural areas close to the mountains”. Note that not only some non-spatial attributes but also the neighbourhood of mountains (after three extensions) are significant for the characterization of the target regions.

community has a high proportion of retired persons \Rightarrow

apartments per building =	very low ($n = 0, f(prop) = 9.1$) \wedge
rate of foreigners =	very low ($n = 0, f(prop) = 8.9$) \wedge
rate of academics =	medium ($n = 0, f(prop) = 6.3$) \wedge
average size of enterprises =	very low ($n = 0, f(prop) = 5.8$) \wedge
object type = mountain	($n = 3, f(prop) = 4.1$)

Figure 7.17. Rules characterizing the target objects of figure 7.16

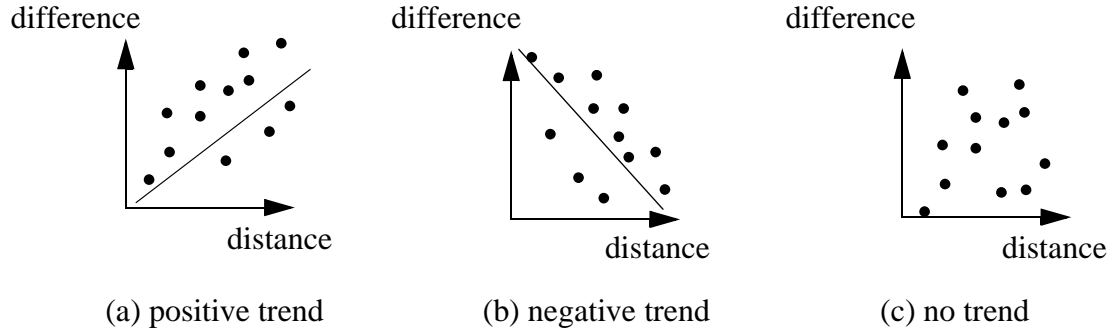


Figure 7.18. Sample linear trends

5 Spatial Trend Detection

We define a *spatial trend* as a regular change of one or more non-spatial attributes when moving away from a given start object o . We use neighbourhood paths starting from o to model the movement and we perform a regression analysis on the respective attribute values for the objects of a neighbourhood path to describe the regularity of change.

5.1 Algorithms

First, we define the task of spatial trend detection for some source object $o_1 \in DB$. To detect regular changes of some non-spatial attributes, we perform a regression analysis as follows. The independent variable (X) yields the distance between any database object o_2 and the source object. The dependent variable (Y) measures the difference of the values of some non-spatial attribute(s) for o_1 and o_2 . Then, the sets X and Y contain one observation for each element of a subset S of DB . If the absolute value of the correlation coefficient is found to be large enough, S identifies a part of DB showing a significant spatial trend for the specified attributes(s) starting from o_1 . In the following, we will use linear regression, since it is efficient and often the influence of some phenomenon to its neighbourhood is either linear or may be transformed into a linear model, e.g. exponential regression. Figure 7.18 illustrates a strong positive (correlation coefficient $\gg 0$) and a strong negative (correlation coefficient $\ll 0$) linear trend as well as a situation where no significant (linear) trend is observed.

In a naive approach, one observation could be made for each element $o_2 \in DB$ and a regression analysis would be conducted once for this whole set of observations. This approach, however, would fail to discover a spatial trend in the following situations:

- a trend is present only locally in the neighbourhood but not globally in the database
- a trend is present not in all directions but only in some directions.

We argue that such situations are very common in real databases. We use the concept of neighbourhood paths to overcome these problems and consider only such objects o_2 which are located on one of the neighbourhood paths starting from o_1 . We stop extending a neighbourhood path as soon as no more significant trend can be found and, thus, provide a means to restrict the search to the neighbourhood of the source object. Furthermore, it is enough that some, not all, neighbourhood paths show a spatial trend of the specified type.

Definition 4: (*spatial trend detection*): Let g be a neighbourhood graph, o an object (node) in g and a be a subset of all non-spatial attributes. Let t be a type of function, e.g. linear or exponential, used for the regression and let $filter$ be one of the filters for neighbourhood paths. Let $min-conf$ be a real number and let $min-length$ as well as $max-length$ be natural numbers. The task of *spatial trend detection* is to discover the set of all neighbourhood paths in g starting from o and having a trend of type t in attributes a with a correlation of at least $min-conf$. The paths have to satisfy the $filter$ and their length must be between $min-length$ and $max-length$.

Definition 7.4 allows different specializations. Either the set of all discovered neighbourhood paths or each of its elements must have a trend of the specified type. For each of these specializations, we present an algorithm to discover such spatial trends. Both algorithms require the same input parameters but they use different methods to search the set of all relevant neighbourhood paths.

The first algorithm discovers *global trends*, i.e. trends for the whole set of all neighbourhood paths with source o having a length in the specified interval. Algorithm *detect-global-trends* performs a breadth-first search of the set of all neighbourhood paths starting from o and it is outlined in figure 7.19.

Beginning from o , the algorithm creates all neighbourhood paths of the same length simultaneously - starting with $min-length$ and continuing until $max-length$. The regression is performed

```

detect-global-trends(graph  $G_r^{DB}$ ; object  $o$ ; attribute  $a$ ; type  $t$ ; real  $min-conf$ ,
                      integer  $min-length, max-length$ ; filter  $f$ )
initialize a list of  $paths$  to the set extensions( $G_r^{DB}$ ,  $\{path(o)\}$ ,  $min-length$ ,  $f$ );
initialize an empty set of  $observations$ ;
initialize  $last-correlation$  and  $last-paths$  as NULL;
initialize  $first-pos$  to 1;
initialize  $last-pos$  to  $min-length$ ;
while  $paths$  is not empty do
    for each  $path$  in  $paths$  do
        for  $object$  from  $first-pos$  of  $path$  to  $last-pos$  of  $path$  do
            calculate  $diff$  as  $a(object) - a(o)$  and calculate  $dist$  as  $dist(object, o)$ ;
            insert the tuple  $(diff, dist)$  into the set of  $observations$ ;
        perform a regression of type  $t$  on the set of  $observations$ ;
        if  $abs(correlation)$  of the resulting regression function  $\geq min-conf$  then
            set  $last-paths$  to  $paths$  and  $last-correlation$  to correlation;
            set  $last-slope$  to the slope of the resulting regression function;
            if  $length(paths) < max-length$  then
                replace the  $paths$  by the set extensions( $G_r^{DB}$ ,  $paths$ ,  $length(paths) + 1$ ,  $f$ );
                increment  $last-pos$  by 1;
                set  $first-pos$  to  $last-pos$ ;
            else set  $paths$  to the empty list;
        else return  $last-paths$ ,  $last-correlation$  and  $last-slope$ ;
return  $last-paths$ ,  $last-correlation$  and  $last-slope$ ;

```

Figure 7.19. Algorithm detect-global-trends

once for each of these sets of all paths of the same length. If no trend of length l with $abs(correlation) \geq min-conf$ is detected, then the path extensions of length $l+1$, $l+2$, . . . , $max-length$ are not created due to efficiency reasons. The parameter $min-length$ has to be chosen large enough because a trend may become significant only at a certain minimum length of a neighbourhood path. The algorithm returns the set of neighbourhood paths of the maximum length having a trend with $abs(correlation) \geq min-conf$. Furthermore, the slope of the regression function and its correlation coefficient are returned. The slope describes the degree of change of the specified attribute when moving away from the source object o : a large slope is found for a quickly changing attribute whereas a small slope indicates that the attribute changes only slowly in the neighbourhood of o . The correlation coefficient measures the confidence of the discovered trend - the larger the correlation coefficient the better can the observations be approximated by the regression function and the higher is the confidence of the discovered trend.

The second algorithm discovers *local trends*, i.e. trends for a single neighbourhood path with source o having a length in the specified interval. Algorithm *detect-local-trends* creates the neighbourhood paths starting from o in a depth-first manner which is presented in figure 7.20. A regression is performed once for each of the neighbourhood paths with $max-length \geq length \geq min-length$ and a path is only extended further if it has a significant trend (i.e. $abs(correlation)$ of the resulting regression function $\geq min-conf$). The algorithm returns two sets of paths showing a significant spatial trend: a first set of positive trends and a second set of negative trends.

```

detect-local-trends(graph  $G_r^{DB}$ ; object  $o$ ; attribute  $a$ ; type  $t$ ; real  $min-conf$ ,
                    integer  $min-length, max-length$ ; filter  $f$ )
initialize a list of paths to the set extensions( $G_r^{DB}$ ,  $path(o)$ ,  $min-length$ ,  $f$ );
initialize two empty sets of positive-trends and negative-trends;
while paths is not empty do
    initialize the set of observations as empty;
    remove the next element of paths and take it as path;
    for object from  $min-length$ -th object of path to last object of path do
        calculate diff as  $a(object) - a(o)$  and calculate dist as  $dist(object, o)$ ;
        insert the tuple (diff, dist) into the set of observations;
    perform a regression of type  $t$  on the set of observations;
    if  $abs(correlation)$  of the resulting regression function  $\geq min-conf$  then
        if  $correlation > 0$  then
            insert the tuple (path, correlation) into the set of positive-trends;
        else
            insert the tuple (path, correlation) into the set of negative-trends;
        if  $length(path) < max-length$  then
            add the extensions( $G_r^{DB}$ , path,  $length(path) + 1$ ,  $f$ ) to the head of paths;
return positive-trends and negative-trends;

```

Figure 7.20. Algorithm detect-local-trends

5.2 Application: Economic Geography

Economic geographers may be interested in discovering spatial trends of various properties of communities such as the economic power. In the following, we illustrate the potential of spatial trend detection on the BAVARIA database which was used for an experimental performance evaluation of the algorithms (Ester et al., 1998).

Spatial trends describe a regular change of non-spatial attributes when moving away from a start object o . The two above algorithms may produce different patterns of change for the same start object o .

The existence of a global trend for a start object o indicates that if considering all objects on all paths starting from o the values for the specified attribute(s) *in general* tend to increase (decrease) with increasing distance. Figure 7.21 (a) depicts the result of algorithm *global-trend* for the attribute “average rent” and the city of Regensburg as a start object.

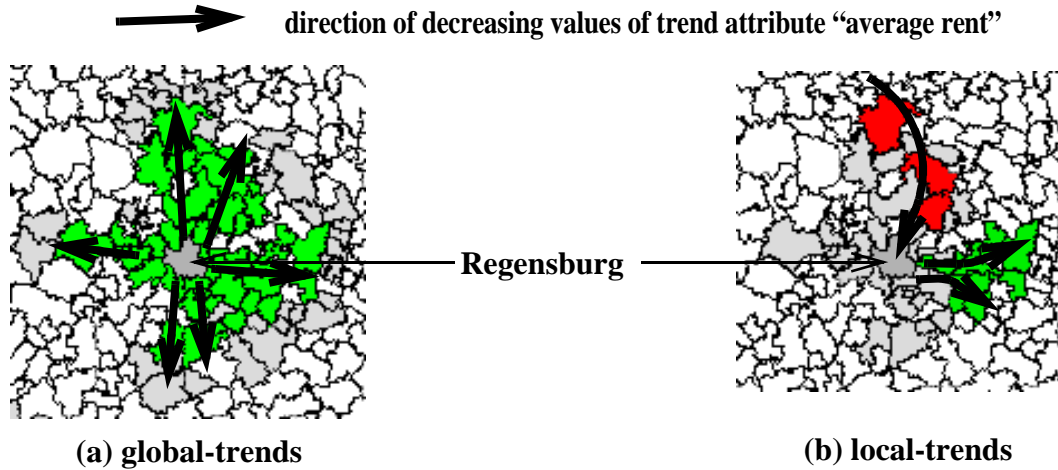


Figure 7.21. Visualization of trends starting from the city of Regensburg

Algorithm *local-trends* detects single paths starting from an object o and having a certain trend. The paths starting from o may show different pattern of change, e.g., some trends may be positive while the others may be negative. Figure 7.21 (b) illustrates this case for the attribute “average rent” and the city of Regensburg as a start object.

Both algorithms were applied to the Bavaria database varying *min-confidence* from 0.6 to 0.8 for the attribute “average rent” and linear type of regression. The predicate *intersects* was used as the neighbourhood relation to define the graph. The filter *vertical starlike* for paths was used because due to our domain knowledge we expected the most significant trends in north-south direction. The length of the paths was restricted by *min-length* = 4 and *max-length* = 7.

The spatial objects within a trend region, i.e. either the start objects or the objects forming the paths, may be the subject of further analysis. For instance, algorithm *global-trend* may detect regions showing a certain global trend, and algorithm *local-trends* then finds within these regions some paths having the inverse trend (see figure 7.21). Then, we may try to find an explanation for those “inverse” paths.

Another possibility of further analysis of the discovered trends is as follows:

- First, detect “centres” for a given attribute, i.e. communities with a significantly decreasing spatial trend of this attribute. Algorithm global-trend can be used for this task.
- Second, apply the spatial characterization to these centres to find their common properties.

Ester et al. (1998) report the results of the following experiment. In a first step, centres for attribute “average rent” were detected: minimum correlation was set to 0.7 and only those communities were selected where the slope of the trend was less than -10^{-4} and the path length was not smaller than 5, i.e. only linear trends that are noticeably decreasing were detected.

With this definition, 24 centres out of the 2043 communities were found. The characterization rule discovered for these centres contains the following properties:

community is a centre \Rightarrow
 rate of academics = high ($n = 1, f(prop) = 9.1$) \wedge
 average number of persons per household = low ($n = 1, f(prop) = 2.5$) \wedge
 rate of foreigners = low ($n = 1, f(prop) = 2.8$)

Note that none of the attributes was significant for $n = 0$, i.e. without considering the neighbourhood of the target object. Only if we extend the target regions by one neighbour, we can see characteristic properties. Thus, this result could not be found by a non-spatial characterization algorithm.

6 Spatial Classification

The task of *classification* is to assign an object to a class from a given set of classes based on the attribute values of the object. In *spatial classification* the attribute values of neighbouring objects may also be relevant for the membership of objects and therefore have to be considered as well.

Ester et al. (1997) proposed a spatial classification algorithm based on the well-known ID3 algorithm (Quinlan, 1986) which was designed for relational databases. The extension to spatial attributes is to consider also the attribute of objects on a neighbourhood path starting from the current object. Thus, we define *generalized attributes* for a neighbourhood path $p = [o_1, \dots, o_k]$ as tuples (attribute-name, index) where index is a valid position in p representing the attribute with attribute-

name of object o_{index} . The generalized attribute (economic-power,2), e.g., represents the attribute economic-power of some (direct) neighbour of object o_I .

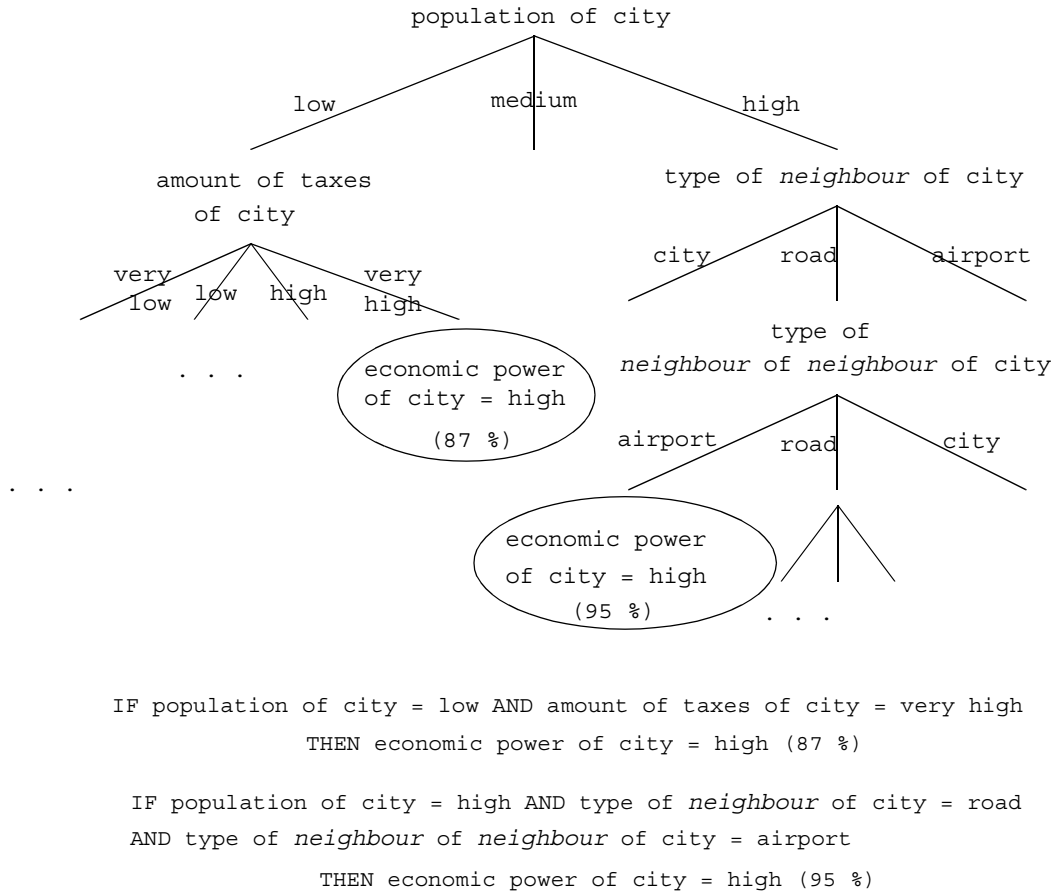


Figure 7.22. Sample decision tree and rules discovered by the classification algorithm

Because it is reasonable to assume that the influence of neighbouring objects and their attributes decreases with increasing distance, we can limit the length of the relevant neighbourhood paths by an input parameter max-length. Furthermore, the classification algorithm allows the input of a predicate to focus the search for classification rules on the objects of the database fulfilling this predicate. Figure 7.22 depicts a sample decision tree and two rules derived from it. Economic power has been chosen as the class attribute and the focus is on all objects of type city.

In figure 7.23, we present the pseudo code of the algorithm for spatial classification which finds all paths from the root to one of the leaves of a decision tree, where all attributes yield an informa-

tion gain of at least ϵ . Note that the algorithm uses a predicate *larger_distance*, i.e. “ $distance(n_1, n_{i+1}) > distance(n_1, n_i)$ ” to restrict the creation of neighbourhood paths.

```

spatial_classification_rules(db:Set_of_Objects; class_attr:Attribute;
                               NPred:NeighbourhoodRelation; max_length:Int)
    NPredGraph:= create the neighbourhood graph for db w.r.t. NPred;
    paths:=extensions(NPredGraph, {db}, max_length, larger_distance);
    classify(class_attr, NPredGraph, EMPTY_RULE, paths, max_length);

classify(class_attr:Attribute; rule:ClassificationRule; paths:set_of_paths; max_length:Int)
    max_info_gain:=0.0;
    max_attr:=NULL;
    for i from 1 to max_length do
        for each generalized attribute (Aj,i) not used in rule do
            info_gain:=calculate_information_gain(Aj, class_attr, i, paths);
            if info_gain > max_info_gain then
                max_attr:=Aj;
                max_neighbours:=i;
                max_info_gain:=info_gain;
    if max_attr  $\neq$  NULL and max_info_gain >  $\epsilon$  then
        for each value of max_attr do
            extended_rule:=rule + “max_attr,max_neighbours,value”;
            classify(class_attr, extended_rule, paths, max_length);

calculate_information_gain(attr,class_attr: Attribute; index:Int; paths:set_of_paths);
    for each path in paths do
        consider attr of the index-th object of path and class_attr of the first object of path
        for the calculation of the information gain

```

Figure 7.23. Algorithm spatial_classification_rules

Another algorithm for spatial classification is presented by Koperski et al. (1998). It works as follows: The relevant attributes are extracted by comparing the attribute values of the target objects with the attribute values of their nearest neighbours. The determination of relevant attributes is based on the concept of the *nearest hit* (the nearest neighbour belonging to the same class) and the *nearest miss* (the nearest neighbour belonging to a different class). In the construction of the decision tree, the neighbours of target objects are not considered individually. Instead, so-called *buffers* are created around the target objects and the non-spatial attribute values are aggregated over all objects contained in the buffer. For instance, in the case of shopping malls a buffer may represent the

area where its customers live or work. The size of the buffer yielding the maximum information gain is chosen and this size is applied to compute the aggregates for all relevant attributes.

Whereas the property of being a nearest neighbour cannot be directly expressed by our neighbourhood relations, it is possible to extend our set of neighbourhood relations accordingly. The proposed database primitives are, however, sufficient to express the creation of buffers for spatial classification by using a distance-based neighborhood predicate.

7 Conclusions

In this chapter, we introduced a database-oriented framework for spatial data mining which is based on the concepts of neighbourhood graphs and paths. A small set of basic operations on these graphs and paths were defined as database primitives for spatial data mining. Furthermore, techniques to efficiently support the database primitives by a commercial DBMS were presented. In the following sections, we covered the main tasks of spatial data mining: spatial clustering, spatial characterization, spatial trend detection and spatial classification. For each of these tasks, we presented algorithms as well as prototypical applications in domains such as the earth sciences and geography. Thus, we demonstrated the practical impact of these algorithms of spatial data mining.

The following issues indicate interesting directions for future research. The database primitives were implemented on top of the commercial DBMS Illustra. Since the system overhead imposed by this DBMS is rather large, techniques of improving the efficiency should be investigated. For example, techniques for processing sets of related *neighbours* operations which provide more information to the DBMS can be used to improve the overall efficiency of mining algorithms using the database primitives.

In some spatial databases the dimension of time plays an important role: the history of the relevant part of the world is stored for the purpose of analysis, for example raster images of the same area of the surface of the earth taken at different times. Data mining in such spatio-temporal databases is a promising area of future research. For example, geographers may be interested in learning spatio-temporal rules describing the process of growth of urban landuse.

References

- Agrawal R., Imielinski T., and Swami A. 1993 "Database Mining: A Performance Perspective", IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 6, pp. 914-925.
- Agrawal R., and Srikant R. 1994 "Fast Algorithms for Mining Association Rules in Large Databases" Proc. 20th Int. Conf. Very Large Data Bases (VLDB '94), pp. 478-499.
- Chen M.-S., Han J., and Yu P. S. 1996 "Data Mining: An Overview from a Database Perspective", IEEE Trans. on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 866-883.
- Egenhofer M. J. 1991 "Reasoning about Binary Topological Relations", Proc. 2nd Int. Symp. on Large Spatial Databases, Zurich, Switzerland, pp. 143-160.
- Ester M., Frommelt A., Kriegel H.-P., and Sander J. 1998 "Algorithms for Characterization and Trend Detection in Spatial Databases", Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, New York City, NY, pp. 44-50.
- Ester M., Frommelt A., Kriegel H.-P., and Sander J. 2000 "Spatial Data Mining: Database Primitives, Algorithms and Efficient DBMS Support", Data Mining and Knowledge Discovery, an International Journal, Kluwer Academic Publishers, Vol. 4, No. 2/3.
- Ester M., Kriegel H.-P., and Sander J. 1997 "Spatial Data Mining: A Database Approach", Proc. 5th Int. Symp. on Large Spatial Databases, Berlin, Germany, pp. 47-66.
- Ester M., Kriegel H.-P., Sander J., and Xu X. 1996 "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. Portland, Oregon, AAAI Press, Menlo Park, California, pp. 226-231.
- Ester M., Kriegel H.-P., Sander J., and Xu X. 1997: "Density-Connected Sets and their Application for Trend Detection in Spatial Databases". Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining. Newport Beach, California, AAAI Press, Menlo Park, California.
- Fayyad U. M. 1997 "Knowledge Discovery in Databases: An Overview", ILP 1997, pp. 3-16.
- Fayyad U. M., Piatetsky-Shapiro G., and Smyth P. 1996 "Knowledge Discovery and Data Mining: Towards a Unifying Framework", Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. Portland, Oregon, AAAI Press, Menlo Park, California, pp. 82 - 88.
- Gueting R. H. 1994 "An Introduction to Spatial Database Systems", Special Issue on Spatial Database Systems of the VLDB Journal, Vol. 3, No. 4.
- Guttman A. 1984 "R-trees: A Dynamic Index Structure for Spatial Searching", Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 47-54.
- Han J., Koperski K., and N. Stefanovic 1997 "GeoMiner: A System Prototype for Spatial Data Mining", Proc. ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD '97), Tucson, AR.

- Koperski K., Adhikary J., Han J. 1996 “Knowledge Discovery in Spatial Databases: Progress and Challenges”, Proc. SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Technical Report 96-08, University of British Columbia, Vancouver, Canada.
- Koperski K., and Han J. 1995 “Discovery of Spatial Association Rules in Geographic Information Databases”, Proc. 4th Int. Symp. on Large Spatial Databases (SSD '95), Portland, ME, pp 47-66.
- Koperski K., Han J., and Stefanovic N. 1998 “An Efficient Two-Step Method for Classification of Spatial Data”, Proc. Symposium on Spatial Data Handling (SDH '98), Vancouver, Canada.
- Lu W., and Han J. 1992 “Distance-Associated Join Indices for Spatial Range Search”, Proc. 8th Int. Conf. on Data Engineering, Phoenix, AZ, pp. 284-292.
- Ng R. T., and Han J. 1994 “Efficient and Effective Clustering Methods for Spatial Data Mining”, Proc. 20th Int. Conf. on Very Large Data Bases, Santiago, Chile, 1994, pp. 144-155.
- Niemann H. 1990 “Pattern Analysis and Understanding”. Springer-Verlag, Berlin.
- Quinlan J.R. 1986 “Induction of Decision Trees”, Machine Learning, Vol.1, pp. 81-106.
- Richards A.J. 1983. “Remote Sensing Digital Image Analysis. An Introduction”. Berlin: Springer Verlag.
- Rotem D. 1991 “Spatial Join Indices”, Proc. 7th Int. Conf. on Data Engineering, Kobe, Japan, pp. 500-509.
- Sander J., Ester M., Kriegel H.-P., and Xu X. 1998 “Density-Based Clustering in Spatial Databases: A New Algorithm and its Applications”, Data Mining and Knowledge Discovery, an International Journal, Kluwer Academic Publishers, Vol.2, No. 2.
- Valduriez P. 1987 “Join Indices“, ACM Transactions on Database Systems, Vol. 12, No. 2, pp. 218-246.