

SDMOQL: An OQL-based Data Mining Query Language for Map Interpretation Tasks

Donato Malerba Annalisa Appice Nicola Vacca

Dipartimento di Informatica, Università degli Studi di Bari
via Orabona 4, 70125 Bari, Italy
{malerba, appice, vacca}@di.uniba.it

Abstract. Spatial data mining denotes the extraction of patterns from both spatial and aspatial data, possibly stored in a spatial database. An important application of spatial data mining methods is the extraction of knowledge from a Geographic Information System. INGENS (Inductive Geographic Information System) is a prototype GIS which integrates data mining tools to assist users in their task of topographic map interpretation. The system can mine geographical concepts that are not explicitly available in the database. The spatial data mining process is aimed at a user who controls the parameters of the process by means of a mining query written in a mining query language. This paper presents SDMOQL, a spatial mining query language, based on the standard OQL, which permits the specification of the task-relevant data, the kind of knowledge to be mined, the background knowledge and the hierarchies, and the interestingness measures. SDMOQL currently supports two data mining tasks in INGENS: inducing classification rules and discovering association rules. Only the part of the language defined for classification tasks is presented in the paper. Some constraints on the query language are identified by the particular mining task. The syntax of the query language is described and the application to a real repository of maps is briefly reported.

1. Introduction

Spatial data are important in many applications, such as computer-aided design, image processing, VLSI, and geographic information systems (GIS). This steady growth of spatial data is outpacing the human ability to interpret them. There is a pressing need for new techniques and tools that find implicit regularities hidden in the spatial data. *Spatial data mining* denotes the extraction of *spatial patterns* from both spatial and aspatial data, possibly stored in a spatial database. Generally speaking, a *spatial pattern* is a pattern showing the interaction of two or more spatial objects or space-dependant attributes according to a particular spacing or set of arrangements [1].

Several works on spatial data mining have already been reported in the literature. An overview of spatial data mining can be found in [9], while seminal work on mining spatial association rules is reported in [8]. New algorithms for spatial classification and spatial trend analysis are illustrated in [3], and a survey on spatial clustering methods in data mining is reported in [7]. A database perspective on spatial data mining is given in the work by Ester *et al.* [4], who define a small set of database

primitives for the manipulation of neighborhood graphs and paths used in some spatial data mining systems. An Inductive Logic Programming (ILP) perspective on spatial data mining is reported in [14], which proposes a logical framework for spatial association rule mining.

GIS offers an important application area where spatial data mining techniques can be effectively used. In the work by Malerba *et al.* [13], it can be seen how some classification patterns, induced from georeferenced data, can be used in topographic map interpretation tasks. A prototype of GIS has been built around this application. INGENS (Inductive Geographic Information System) [12] is an innovative GIS with data mining facilities used to support sophisticated end users in their topographic map interpretation tasks. In INGENS, each time a user wants to query the database on some geographical objects not explicitly modeled, he/she can prospectively train the system to recognize such objects and to create a special user view. Training is based on a set of examples and counterexamples of geographic concepts of interest to the user (e.g., ravine or steep slopes). Such concepts are not explicitly modeled in the map legends, so they cannot be retrieved by simple queries. Furthermore, the user has serious difficulty formalizing their operational definitions. Therefore, it is necessary to rely on the support of a knowledge discovery system that generates some plausible "definitions". The sophisticated user is simply asked to provide a set of (counter-) examples (e.g., map cells) and a number of parameters that define the data mining task more precisely.

An INGENS user should not suffer from problems related to the integration of different technologies, such as data mining, OODBMS, and GIS. In general, to solve these problems the use of *data mining query languages* (data mining language, for short) has been proposed, which interface users with the whole system and hide the different technologies [6]. A data mining language allows a user to formulate a data mining task without paying attention at logical and physical representation problems, as well as the correct procedural order in which some data mining steps should be performed. A data mining language is as important for quickly developing decision support applications as SQL for quickly implementing business applications. Interpreters for a data mining language are the gate through the pattern catalogue and the data mining processor, just as the query language interpreters are the gate through the system catalogue and the run-time database processor. A casual user can find patterns by means of a data mining language in the same way he/she can find data by means of a SQL query, without developing an *ad hoc* application which satisfies his/her information need. Furthermore, having a data mining query language provides a foundation on which graphical user interfaces can be built.

A data mining query language must incorporate a set of data mining *primitives* designed to facilitate efficient, fruitful knowledge discovery. Such primitives include the specification of the portions of the database in which the user is interested, including the database attributes or data warehouse dimensions of interest, the kinds of knowledge to be mined, background knowledge useful in guiding the discovery process, interestingness measures of pattern evaluation, and how the discovered knowledge should be visualized.

The problem of designing a spatial mining language has received little attention in the literature. Koperski designed GMQL (Geo Mining Query Language)

[10], a language for the formulation of the input to data mining processes. The language uses a unified syntax, which allows a variety of data mining queries. GMQL is based on DMQL (Data Mining Query Language) [5], which was developed for

mining knowledge from *relational* databases. SQL remains the milestone on which both data mining languages are built.

This paper presents SDMOQL (Spatial Data Mining Object Query Language) a spatial mining query language for INGENS sophisticated users. The presence of a spatial mining query language in a system like INGENS facilitates the expert user who has to train the system by means of inductive queries. Since the Map Repository is implemented in a commercial OODBMS, INGENS spatial mining language is based on OQL, the standard defined by ODMG (Object Database Management Group) for designing object oriented models. The paper is organized as follows. INGENS architecture and conceptual database schema are described in the next section, while in Section 3 the spatial data mining process in INGENS is introduced. In Section 4 SDMOQL syntax is presented. Finally, in Section 5, a complete example of SDMOQL use in INGENS is described.

2. INGENS architecture and conceptual database schema

The architecture of INGENS is illustrated in Fig. 1. The interface layer implements a graphical user interface (GUI), which is a Java applet. The layer of the application enablers makes several facilities available to INGENS users. In particular, the *Map Descriptor* is the application enabler responsible for the automated generation of first-order logic descriptions of some geographical objects. The *Data Mining Server* provides a suite of data mining systems that can be run concurrently by multiple users to train INGENS. The *Query Interpreter* allows any user to formulate queries in SDMOQL language. Therefore, it is the responsibility of the Query Interpreter to select the involved objects from the Map Repository, to ask the Map Descriptor to generate their logical descriptions and to invoke the Data Mining Server to train the system. The *Map Converter* is a suite of tools which support the acquisition of maps from external sources, namely raster images from scanners and geographical objects

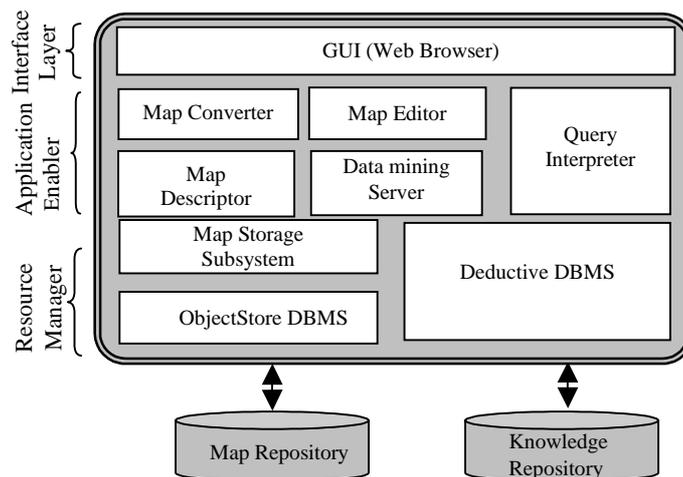


Fig. 1. INGENS three-layered software architecture.

from vectorized maps in a proprietary format. The *Map Editor* permits the integration and/or modification of information acquired by means of the *Map Converter*. The lowest layer manages resources like the *Knowledge Repository* and the *Map Repository*. The former contains the geographical concepts induced by the Data Mining Server. The *Map Repository* is the database instance that contains the actual collection of maps stored in the GIS. The *Map Storage Subsystem* is involved in storing, updating and retrieving items in and from the map collection. As a *resource manager*, it represents the only access path to the data contained in the Map Repository and which are accessed by multiple, concurrent clients. Geographic data are organized according to an object-oriented data model. At the *conceptual* level, the model is described by the class diagram in Fig. 2.

Each map is stored according to a hybrid tessellation – topological model. The tessellation model follows the usual topographic practice of superimposing a regular grid on a map in order to simplify the localization process. Indeed each map in the repository is divided into square cells of the same size. In the topological model of each cell it is possible to distinguish two different structural hierarchies: *physical* and *logical*. The physical hierarchy describes the geographical objects by means of the most appropriate physical entity, that is: point, line or region. Some topological relationships between points, lines and regions are modelled in the conceptual design, namely points inside a region or on its border, and regions disjointing/meeting/overlapping/containing/equalling/covering other regions. The logical hierarchy expresses the semantics of geographical objects, independent of their physical representation. Since the conceptual data model has been designed to store topographic maps, the entity *logical_object* is a total generalization of eight distinct entities, namely hydrography, orography, land administration, vegetation, administrative (or political) boundary, ground transportation network, construction and built-up area, which represent different geographic layers in a topographic map. A geographic layer describes one or more geographical objects of the same class.

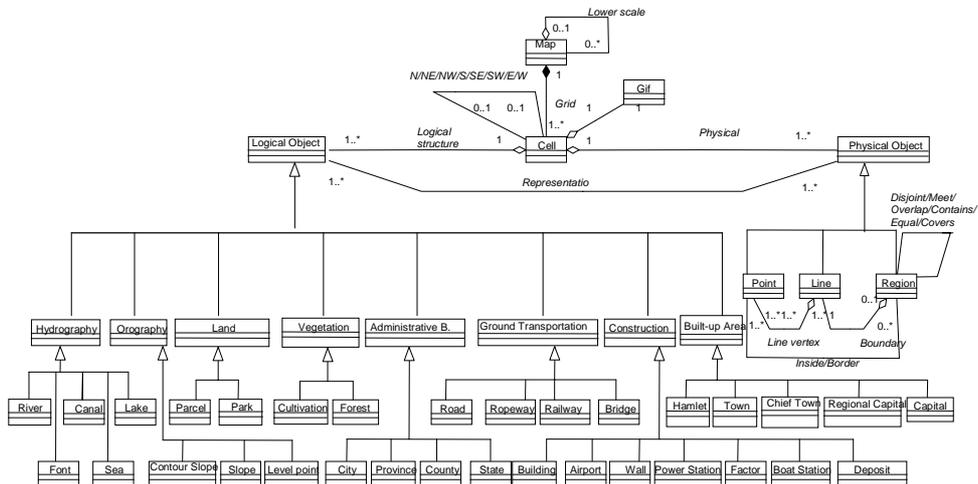


Fig. 2. Class diagram of INGENS conceptual model in Unified Modeling Language (UML).

3. Spatial Data Mining process in INGENS

The spatial data mining process in INGENS is aimed at a user who controls the parameters of the process. Initially, the query written in SDMOQL is syntactically and semantically analyzed. Then the Map Descriptor generates a highly conceptual qualitative representation of the raw data stored in the object-oriented database. This representation is a conjunctive formula in a first-order logic language, whose atoms have the following syntax:

$$f(t_1, \dots, t_n) = value,$$

where f is a function symbol called *descriptor*, t_i are terms and the *value* is taken from the range of f . Descriptors can be either nominal or linear according to the ordering relation defined on its range. A set of descriptors used in INGENS is reported in Table 1. It is quite general and it can capture geometric, topological and directional features of a geographical object in a map. The operational semantics of these descriptors is based on a set of methods defined in the object-oriented model of the Map Repository.

The following is an example of the qualitative representation of river and road objects by means of spatial and aspatial descriptors :

```
contain(x1,x6)=true, contain(x1,x11)=true,
contain(x1,x41)=true, type_of(x1)=cell,
type_of(x6)=road, type_of(x11)=river,
type_of(x41)=river, subtype_of(x6)=cart_track_road,
color(x6)=black, color(x11)=blue, color(x41)=blue,
extension(x6)=747.976, extension(x11)=1131.84,
extension(x41)=850.423, line_shape(x6)=curvilinear,
line_shape(x11)=curvilinear,
line_shape(x41)=curvilinear,
line_to_line(x11,x41)=almost_parallel.
```

This qualitative data representation can be easily translated into Datalog with built-in predicates [2]. Thanks to this transformation, it is possible to use the output of the Map Descriptor module in many data mining algorithms designed to run with Datalog-like input.

Data mining algorithms mine spatial patterns based on the qualitative descriptions of the extracted data. Finally, the results of the mining process are presented to the user. The graphical feedback is very important in the analysis of the results.

4. Design of a data mining language for INGENS

SDMOQL is designed to support the interactive data mining process in INGENS. Designing a comprehensive data mining language is a challenging problem because data mining covers a wide spectrum of tasks from data classification to mining association rules. The design of an effective data mining query language requires a deep understanding of the power, limitation and underlying mechanisms of the various kinds of data mining tasks. We looked at primitives for defining a data mining task in the form of a data mining query. The primitives specify the following:

Table 1. Set of descriptors extracted by the Map Descriptor module in INGENS.

Feature	Meaning	Type	Domain	
			Type	Value
Contain(X,Y)	Cell X contains object Y	Topological relation	boolean	{true, false}
Type_of(Y)	Type of Y	Aspatial attribute	nominal	33 nominal values
Subtype_of(Y)	Specialization of the type of Y	Aspatial attribute	nominal	101 nominal values
Color(Y)	Color of Y	Aspatial attribute	nominal	{blue, brown, black}
Area(Y)	Area of Y	Geometrical attribute	linear	[0..MaxArea]
Density(Y)	Density of Y	Geometrical attribute	ordinal	Symbolic names chosen by expert user
Extension(Y)	Extension of Y	Geometrical attribute	linear	[0..MaxExt]
Geo_direction (Y)	Geographic direction of Y	Directional attribute	nominal	{north, east, north_west, north_east}
Line_shape(Y)	Shape of the linear object Y	Geometrical attribute	nominal	{straight, curvilinear, cuspidal}
Altitude(Y)	Altitude of Y	Geometrical attribute	linear	[0.. MaxAlt]
Line_to_line (Y,Z)	Spatial relation between two lines Y and Z	Hybrid relation	nominal	{almost parallel, almost perpendicular}
Distance(Y,Z)	Distance between two lines Y and Z	Geometrical relation	linear	[0..MaxDist]
Region_to_region (Y,Z)	Spatial relation between two regions Y and Z	Topological relation	nominal	{disjoint, meet, overlap, covers, covered_by, contains, equal, inside}
Line_to_region (Y,Z)	Spatial relation between a line Y and a region Z	Hybrid relation	nominal	{along_edge, intersect}
Point_to_region (Y,Z)	Spatial relation between a point Y and a region Z	Topological relation	nominal	{inside, outside, on_boundary, on_vertex}

- The set of task-relevant data to be mined
- The kind of knowledge to be mined
- The background knowledge to be used in the discovery process
- The interestingness measures and thresholds for pattern evaluation
- The expected representation for visualizing the discovered patterns.

In Sections 4.1 to 4.4 the syntax for the first four data mining primitives is both formally specified in extended BNF and explained through various examples of possible mining problems.

4.1 Syntax for task-relevant data specification

The first step in defining a data mining task is the specification of the data on which mining is to be performed. In traditional data mining applications, it is generally sufficient to specify database attributes, data warehouse dimensions or cubes which contain the portions of data of interest for two reasons:

1. no interaction between objects is assumed, so that each object can be effectively described by a tuple in a relation
2. no complex transformation of stored data is required.

In this case, relatively simple SQL queries can be used to specify the data set to mine.

On the contrary, in GIS applications, attributes of the neighbors of some spatial object of interest may influence the object itself and therefore they should be considered as well. In fact, the explicit location and extension of spatial objects define implicit relations of a spatial neighborhood (such as topological, distance and direction relations), which cannot be neglected by spatial data mining algorithms. Consequently, the data set to mine cannot be straightforwardly represented by means of a relational table, where distinct tuples refer to distinct objects. Moreover, working at the level of stored data, that is geometric representations (points, lines and regions) of geographical objects, is often undesirable. The GIS user is interested in working at higher conceptual levels, where human-interpretable properties and relations between *geographical objects* are expressed. A typical example is represented by the possible relations between two roads, which either cross each other, or run parallel, or can be confluent, independently of the fact that the two roads are represented by one or more tuples of a relational table of “lines” or “regions”. Therefore, complex transformations are required to describe geographical objects to be mined.

To solve these problems, the SDMOQL interpreter allows users to select the geographical objects that are relevant to the data mining task, and then it invokes the Map Descriptor in order to produce their high level conceptual descriptions, including both properties and relations.

The selection of geographical objects is performed by means of simplified OQL queries with a SELECT-FROM-WHERE structure. To explain the constraints imposed on the OQL syntax by this particular application, some examples are reported in the following:

Example 1 - Cell-level query

The user selects *cell 26* from the topographic map of Canosa (Apulia) and the Map Descriptor generates the description of all the objects in this cell.

```

SELECT x FROM x in Cell
WHERE x->num_cell = 26
AND x->part_map->map_name = "Canosa"

```

Example 2 - Layer-level query

The user selects the *layer Horography* from the topographic map of Canosa and the *layer Construction* from any map. The Map Descriptor generates the description of the objects in these layers.

```

SELECT x, y FROM x in Horography, y in Construction
WHERE x->part_map->map_name = "Canosa"

```

Example 3 - Object-level query

The user selects the *objects* of the logic class River and the objects of type motorway (instances of the class Road), from cell 26 of the topographic map of Canosa. The Map Descriptor generates the description of these objects.

```

SELECT x, y FROM x in River, y in Road
WHERE x->part_map->map_name = "Canosa" AND
      y->part_map->map_name = "Canosa" AND
      x->log_incell->num_cell = 26 AND
      y->log_incell->num_cell = 26 AND
      y->type_road = "motorway"

```

The above queries do not present semantic problems. However, the next example is an OQL query which is syntactically correct but selects data that cannot be a valid input to the Map Descriptor.

Example 4 - Semantically ambiguous query:

```

SELECT x, y
FROM x in Cell, y in River
WHERE x->num_cell = 26
AND y->log_incell->num_cell = 26

```

This query selects the object *cell 26* and all rivers in it. However, it is unclear whether the Map Descriptor should describe the entire *cell 26* or only the rivers in it, or both. In the first case, a cell-level query must be formulated (see example 1). In the second case, an object-level query produces the desired results (see example 3). In the (unusual) case that both kinds of descriptions have to be generated, the problem can be solved by the UNION operator, applied to the cell-level query and the object-level query. Therefore, the following constraint is imposed on SDMOQL: *the selected data must belong to the same symbolic level (cell, layer or logic object)*. More formally the FROM clause can contain either a group of *Cells* or a set of *Layers*, or a set of *Logic Objects*, but never a mixture of them.

Example 5 - Attributes in the SELECT clause

This example is useful to present the constraints imposed on the SELECT clause.

```

SELECT x.name_river
FROM x in River

```

The query selects the names of all the rivers stored in the database. The result set contains attributes and not geographical objects to be described by a set of attributes and relations. In order to select proper input data for the Map Descriptor, the SELECT clause should return objects of a class in the database schema corresponding to a cell, a layer or a type of logical object. It might be observed that the presence of an attribute in the SELECT clause can be justified when its type corresponds to a class. For instance, the following query:

```
SELECT x->River FROM x in Cell
WHERE x->num_cell=26
```

concerns all rivers in cell 26. Nevertheless, thanks to inverse relations (inverse members) characterizing an object model, it is possible to reformulate it as follows:

```
SELECT x FROM x in River
WHERE x->log_incell->num_cell=26
```

In this way, all the above constraints should be respected.

4.2 The kind of knowledge to be mined

The kind of knowledge to be mined determines the data mining task in hand. For instance, classification rules or decision trees are used in classification tasks, while association rules or complex correlation coefficients are extracted in association tasks. Currently, SDMOQL supports the generation of either classification rules or association rules, which means that only two different mining problems can be solved in INGENS: the former has a predictive nature, while the latter is descriptive. The top-level syntax is defined below:

```
<SDMOQL> ::= <SDMOQL_Statement>; {<SDMOQL_Statement>}
<SDMOQL_Statement> ::= <Spatial_Data_Mining_Statement>
<Spatial_Data_Mining_Statement> ::=
    <Limited_OQL_Query>
    mine
    <Kind_of_Pattern>
<Kind_of_Pattern> ::=
    <Classification_Rules> | <Association_Rules>
```

In particular <Classification_Rules> specifies that patterns for data classification tasks are to be mined:¹

```
<Classification_Rules> ::=
    classification as <Pattern_Name>
    for <Classification_Concept>{,<Classification_Concept>}
    [<Analyze_List>]
<Analyze_List> ::=
    analyze <Descriptor_List>
```

¹ The syntax for association rules is omitted due to space constraints.

In a classification task, the user may be interested in inducing a set of classification rules for a *subset* of the classes (or concepts) to which training examples belong. Typically, the user specifies both “positive” and “negative” examples, that is he/she specifies examples of two different classes, but he/she is interested in classification rules for the “positive” class alone. In this case, the subset of interest for the user is specified in the *<Classification_Concept>*. The **analyze** clause indicates that the descriptions of selected data is based on the spatial/aspatial descriptors in the *<Descriptor_List>*. An example of a classification task activated by an SDMOQL statement is the following:

Example 6 - Classification task

```
SELECT x FROM x in Cell
WHERE x->num_cell >= 5 AND x->num_cell <= 12
mine classification as MorphologicalElements
for class(_)=system_of_farms,class(_)=fluvial_landscape
analyze contain/2, type_of/1, subtype_of/1, area/1,
density/1, extension/1, line_shape/1,
geographic_direction/1, line_to_line/2,
distance/2, line_to_region/2,
region_to_region/2,point_to_region/2
```

In this case, the Map Descriptor generates a symbolic description of the cells with number identifiers between 5 and 12 using the predicates listed in the analyze clause. There are two concepts to be learned: *class(_)=system_of_farms* and *class(_)=fluvial_landscape*. Here the function symbol *class* is unary and “_” denotes the anonymous variables *à la* Prolog. The user can provide examples of these two classes, as well as of other classes. Examples of systems of farms are considered as positive for the first concept in the list and negative for the second concept. The converse is true for examples of fluvial landscapes. Examples of other classes are considered as counterexamples of both classes, for which rules will be generated.

4.3 Syntax for Background Knowledge and Concept Hierarchy Specification

Background knowledge is information provided by a domain expert about the domain to be mined. It can be useful in the discovery process. A top-level syntax is defined below for background knowledge and concept hierarchy specification:

```
<SDMOQL_Statement> ::=
    <Spatial_Data_Mining_Statement>
    |<Background_Knowledge>
    |<Hierarchy>
<Spatial_Data_Mining_Statement> ::=
    <Limited_OQL_Query>
    mine
    <Kind_of_Pattern>
    <Background_Knowledge>
    <Hierarchy>
```

In INGENS, the background knowledge is expressed as a set of definite clauses:

define knowledge *definite_clauses*

Alternatively, the user can specify a set of rules explicitly stored in a deductive database and possibly mined in a previous step:

use background knowledge of users

Username1, Username2 about Predicate_Name1/Argument_Number1,

Username3 about Predicate_Name2/Argument_Number2

An example of background knowledge provided by a user in a geographic mining task is the following:

Example 7 – Defining the concept close_to to support spatial qualitative reasoning

define knowledge

close_to(X,Y)=true:-region_to_region(X,Y)=meet.

close_to(X,Y)=true:- close_to(Y,X)=true.

Concept hierarchies allow knowledge mining at a multiple abstraction levels. They can be used in *roll-up* and *drill-down* operations. Patterns can be rolled up, or viewed at a more general level, by climbing up the concept hierarchy of an attribute, replacing a lower level concept by a higher level one. Patterns can also be drilled down by stepping down the concept hierarchy of an attribute. In order to accommodate the different viewpoints of users with regard to the data, there may be more than one concept hierarchy per attribute or dimension. For instance, some users may prefer to organize census districts (or enumeration districts) by wards and districts, while others may prefer to organize them according to their main purpose (industrial area, residential area, and so on).

There are four major types of concept hierarchies [5]:

- *Schema hierarchies*: belong to total or partial order among attributes in the database schema.
- *Set-grouping hierarchies*: organize values for given attributes or dimensions into groups of constants or range values.
- *Operation-derived hierarchies*: are based on operations specified by experts, or data mining systems.
- *Rule-based hierarchies*: occur when either a whole concept or a portion of it is defined by a set of rules.

In SDMOQL a specific syntax is defined for the first two types of hierarchies. Rule-based hierarchies can be defined as background knowledge. The following examples show how to define a schema hierarchy and a set-grouping hierarchy.

Example 8 - Definition of a schema hierarchy

A user defines a schema hierarchy for a relation activity, as shown in Fig. 3a

define hierarchy Activity **as**

level1:{business_activity, other_activity}

< level0: Activity;

level2:{low_business_activity,high_business_activity}

< level1: business_activity;

This kind of hierarchy is used to mine multi-level spatial association rules [14].

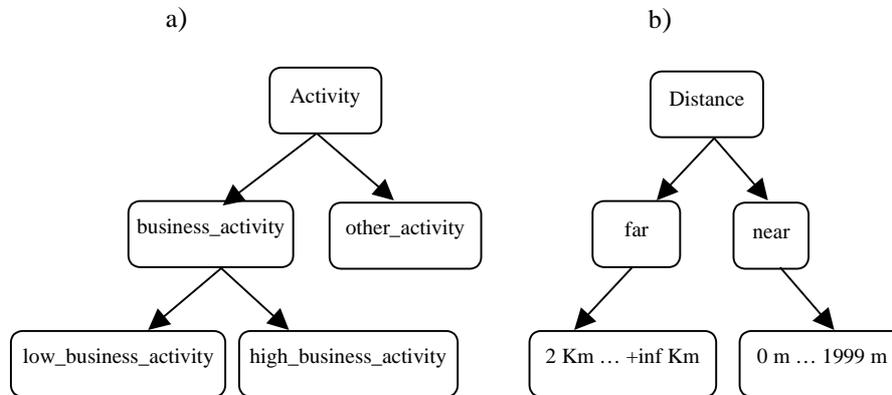


Fig. 3 – Schema hierarchy (a) and set-grouping hierarchy (b).

Example 9 - Definition of a set-grouping hierarchy

The set-grouping hierarchy for *distance* (see Fig. 3b) can be defined in terms of ranges, as follows:

```

define hierarchy Distance for distance/2 as
  level1:{far, near} < level0: Distance;
  level2:{0, 1999} < level1: near;
  level2:{2000, +inf} < level1: far;

```

As in the case of the background knowledge, the following clause:

```

use hierarchy Activity of user Username

```

permits users to import a concept hierarchy defined by another user.

4.4 Syntax for Interestingness Measure Specification

The user can control the data mining process by specifying measures of pattern interestingness and their corresponding thresholds. The SDMOQL top-level syntax is extended as follows:

```

<Spatial_Data_Mining_Statement> ::=
  <Limited_OQL_Query>
  mine
  <Kind_of_Pattern>
  <Background_Knowledge>
  <Hierarchy>
  with <Interestingness_Measures>

```

Interestingness measures may include: threshold values, search biases in the hypotheses space and algorithm-specific parameters. The user can set thresholds such as *confidence*, *support* or *number of learned concepts* as follows:

```

ThresholdParameter threshold Value

```

The user can also bias the search in the hypotheses space by a number of preference criteria, such as maximization of the number of covered examples or minimization of the number of variables in the body of a learned clause, according to the following syntax:

preference criteria (minimize | maximize) Criterion with tolerance Value.

Finally, the user can set the value of a generic input parameter of a data mining algorithm with the statement:

ParameterName = Value

An example of parameters defined for a classification task is reported in the next section.

5. Mining classification rules for topographic map interpretation

In the previous section, the syntax of SDMOQL has been partially defined. Here we present the full specification of a SDMOQL query for the problem of mining classification rules for topographic map interpretation. Let us suppose that a GIS user needs to localize a “*sistema poderal*” (system of farms) in the large territory of his/her interest. This geographical object is not present in the GIS model, thus, only the specification of its operational definition will allow the GIS to find cells containing a system of farms in a vectorized map. Who can provide such a definition? The user is not able to do so for a number of reasons.

Firstly, providing the GIS with operational definitions of some environmental concepts is not a trivial task. For example, the general description of a road given by an expert is “a consolidated way, in the first place used for motor vehicle traffic, including over- and underpasses. Moreover, dividing strips and roadsides [*omissis*] belong to roads.” This declarative, abstract definition is difficult to compile into a query on a map repository.

Secondly, the operational definitions of some geographical objects are strongly dependent on the data model that is adopted by the GIS. For instance, finding relationships between the density of vegetation and climate is easier with a *raster data model*, while determining the preferred orientation of some morphological elements is simpler in a *topological data model*.

Thirdly, different applications of a GIS will require the recognition of different geographical elements in a map. Providing the system in advance with all the knowledge required for its various application domains is simply illusory, especially in the case of wide-ranging projects such as those set up by governmental agencies.

A solution to these problems can be found in the application of data mining techniques. For instance, an INGENS user can train the system to recognize cells with systems of farms, by performing the following SDMOQL query:

```
SELECT x FROM x in Cell
WHERE(x->num_cell >= 1 AND x->num_cell <= 6)
      OR x->num_cell = 11 OR x->num_cell = 34
      OR (x->num_cell >= 15 and x->num_cell <= 17)
mine classification as MorphologicalElements
for class(X)=system_of_farms
analyze contain/2, type_of/1, subtype_of/1, color/1,
         altitude/1, area/1, density/1, extension/1,
         line_shape/1, geographic_direction/1,
         line_to_line/2, distance/2, line_to_region/2,
         region_to_region/2, point_to_region/2
```

with preference criteria

minimize negative_example_covered **with tolerance** 0.6,

maximize positive_example_covered **with tolerance** 0.4,

minimize cost **with tolerance** 0.4

number_of_rules **threshold** 15, consistent **threshold** 500

In this query all the descriptors defined in Table 1 are used to generate the symbolic descriptions of the selected cells. The SDMOQL interpreter analyzes the query and verifies its syntactic and semantic correctness. Then the Map Descriptor generates the symbolic description of the specified cells (see Fig 4) and the expert associates each symbolic description with a concept, in order to define the training



Fig. 4 - Raster and vector representation (a) and symbolic description of cell 11 (b). The cell is an example of a territory where a system of farms is present. The cell is extracted from a topographic chart (Canosa di Puglia 176 IV SW - Series M891) produced by the Italian Geographic Military Institute (IGMI) at scale 1:25,000 and stored in INGENS.

set. Association is made by *binding* variable terms of one of the two concepts *class(X)=system_of_farms* (to be learned) and *class(X)=other* to constant terms in the descriptions of map cells. This step is necessary to create the training set of positive and negative examples for the data mining algorithm ATRE [11], which is used in INGENS for classification tasks. Suppose that the user recognizes cells 5, 6, 11 and 34 as positive examples of a system of farms, he will associate them with the concept *class(X)=system_of_farms* by binding variable *X* to the constant *x1*, which typically represents the whole cell in their corresponding descriptions. Similarly, the user will associate the remaining cells with the concept *class(X)=other*. This binding function is supported by INGENS GUI. The training set obtained is input to ATRE, which returns the classification rules. With reference to the above query, ATRE generates the following clause:

```
class(X1)=system_of_farms :- type_of(X1)=cell,
    contain(X1,X2)=true,type_of(X2)=parcel,
    area(X2) in [67225 .. 187525],
    region_to_region(X2,X3)=meet,
    type_of(X3)=parcel,
    region_to_region(X2,X4)=disjoint,
    region_to_region(X3,X4)=disjoint,
    line_to_region(X5,X4)=adjacent,
    point_to_region(X6,X4)=inside,
    density(X2)=medium.
```

This classification rule can be interpreted as follows: a cell is an example of a system of farms if it contains two adjacent parcels (X2, X3), such that one (X2) has an area between 67,225 and 187,525 square meters and a medium vegetation density, and another geographical object (X4) (possibly a parcel), not overlapping the previous two, but adjacent to a line (X5) (possibly a cart track), contains another geographical object (X6) (possibly an artesian well).

Operational definitions like that reported above can be used either to retrieve new instances of the learned concepts from the Map Repository, or to facilitate the formulation of a query involving geo-referenced abstract concepts not in map legends.

6. Conclusions

In this paper, a spatial data mining language for a prototypical GIS with knowledge discovery facilities has been partially presented. This language is based on a simplified OQL syntax and is defined in terms of the five data mining primitives. For a given query, these primitives define the relevant data for the task, the kind of knowledge to be mined, the background knowledge definition and the concept hierarchies, interestingness measures to be used and the representation forms for pattern visualization. An interpreter for this language has been developed in the system INGENS. It interfaces a Map Descriptor module that can generate a first-order logic description of selected geographical objects. A full example of query formulation and results has been reported for a classification task used in the qualitative interpretation of topographic maps. An extension of this language to other spatial data mining tasks supporting quantitative interpretation of maps is planned for the near future.

References

- [1] DeMers, M.N. (2000). *Fundamentals of Geographic Information Systems*. 2nd ed., John Wiley & Sons.
- [2] Esposito, F., Malerba, D. & Lisi, F.A. (2000). Induction of recursive theories in the normal ILP setting: issues and solutions, in J. Cussens and A. Frisch (Eds.) *Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, 1866, 93-111, Springer, Berlin, Germany.
- [3] Ester, M., Frommelt, A., Kriegel, H. P., Sander, J. (1998). Algorithms for Characterization and Trend Detection in Spatial Databases, *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining*, 44-50, New York City, NY.
- [4] Ester, M., Gundlach, S., Kriegel, H.P., Sander, J. (1999). Database Primitives for Spatial Data Mining, *Proc. Int. Conf. on Database in Office, Engineering and Science*, (BTW '99), Freiburg, Germany.
- [5] Han, J., Fu, Y., Wang, W., Koperski, K. and Zaïane, O. R. (1996). DMQL: a data mining query language for relational databases. In *Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery*, 27-34, Montreal, QB.
- [6] Han, J., Kamber, M. (2000). *Data Mining*, Morgan Kaufmann Publishers. Cap. 4.
- [7] Han, J., Kamber, M. & Tung, A.K.H. (2001). Spatial clustering methods in data mining. In H. J. Miller & J. Han (Eds.), *Geographic Data Mining and Knowledge Discovery*, 188-217, Taylor and Francis, London, UK.
- [8] Koperski, K. & Han, J. (1995). Discovery of Spatial Association Rules in Geographic Information Database. In *Advances in Spatial Database, Proceedings of 4th Symposium, SSD '95*. (Aug. 6-9. Portland, Maine), 47-66, Springer-Verlag, Berlin.
- [9] 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.
- [10] Koperski, K. (1999). A progressive refinement approach to spatial data mining. M. Sc. Warsaw University of technology.
- [11] Malerba, D., Esposito, F., Lisi, F.A. (1998) Learning recursive theories with ATRE. In: Prade, H. (ed.): *Proc. 13th European Conference on Artificial Intelligence*, 435-439, John Wiley & Sons, Chichester, England.
- [12] Malerba, D., Esposito, F., Lanza, A., Lisi, F.A. (2000). Discovering geographic knowledge: The INGENS system. In Ras, Z.W., Ohsuga, S. (Eds.): *Foundations of Intelligent Systems, LNAI 1932*, 40-48, Springer-Verlag, Berlin.
- [13] Malerba, D., Esposito, F., Lanza, A. & Lisi, F.A. (2001). Machine learning for information extraction from topographic maps. In H. J. Miller & J. Han (Eds.), *Geographic Data Mining and Knowledge Discovery*, 291-314, Taylor and Francis, London, UK.
- [14] Malerba, D., Lisi, F.A. (2001). An ILP method for spatial association rule mining. In A. Knobbe and D. van der Wallen (Eds.), *Notes of the ECML/PKDD 2001 Workshop on Multi-Relational Data Mining*, 18-29, Freiburg, Germany.