Classification in Geographical Information Systems

Salvatore Rinzivillo and Franco Turini

Dipartimento di Informatica,
v. Buonarroti, 2 - 56125 Pisa
University of Pisa, Italy
{rinziv,turini}@di.unipi.it

Abstract. The paper deals with the problem of knowledge discovery in spatial databases. In particular, we explore the application of decision tree learning methods to the classification of spatial datasets. Spatial datasets, according to the Geographic Information System approach, are represented as stack of layers, where each layer is associated with an attribute. We propose an ID3-like algorithm based on an entropy measure, weighted on a specific spatial relation (i.e. overlap). We describe an application of the algorithm to the classification of geographical areas for agricultural purposes.

1 Introduction

Spatial data are usually handled by means of a Geographical Information System, that is a system able to represent a territory along with its characteristics. In general a territory is represented as a stack of layers. Each of the layers is associated to a specific attribute, and the layer represents a partitioning of the territory according to the values of the attribute itself. In other words each of the partitions of the layer corresponds to an area of the territory associated to a specific value of the attribute.

Geographical Information Systems provide the user with the possibility of querying a territory for extracting areas that exhibit certain properties, i.e. given combinations of values of the attributes. Just as it is intuitive to extend standard database query language to embody inductive queries, we believe that an analogous approach can be explored for Geographical Information Systems, and, in general, for spatial databases. For example, finding a classification of areas for a given exploitation, agriculture say, or finding associations among areas, or even clustering areas seem to be sensible questions to answer. All the inductive algorithms for knowledge discovery in databases starts from considering a collection of "transactions", usually represented as a table of tuples. In the case of a spatial database, the notion of transaction naturally corresponds to the tuple of attribute values that characterize an area. That leaves us with two problems:

1. the selection of the transaction: i.e. which area with which values;
2. the exploitation of the area in the inductive algorithms.

In this paper we address the problem of constructing a classification tree for classifying areas with respect to their properties. Given spatial databases that store a collection of layers for one or more territories, choose one of the layers as the output class and try to construct a classification tree capable of placing an area in the right class given the values of the other attributes with respect to the area. As a running example, we consider an agricultural territory, and we choose the kind of crop as the classification task, whereas the other attributes concern properties of the ground (amount of water etc.). We show how a suitably adapted ID3 algorithm can do the job. The crucial point is the introduction of a suitable entropy measure, that takes into account specific spatial relations.

In the rest of the section we briefly overview the work related to our research. In Section 2 we formalize the problem and in Section 3, the kernel of the paper, we discuss the algorithm both formally and informally by using the agriculture related running example. The Conclusions section is devoted to discussing our current and future work.

Knowledge Discovery in Spatial Databases. In the analysis of geographically referenced data it is important to take into account the spatial component of the data (i.e. position, proximity, orientation, etc.). Many methods have been proposed in the literature to deal with this kind of information.

Some techniques consider spatial data independently from non-spatial data during the learning step, and they relate them during pattern analysis. For example, in [2] an integrated environment for spatial analysis is presented, where methods for knowledge discovery are complemented by the visual exploration of the data presented on the map. The aim of map visualization is to prepare (or select) data for KDD procedures and to interpret the results. In [1], this environment is used for analyzing thematic maps by means of C4.5 [14] (for pattern extraction from non-spatial data). However, the iterative methods presented there keep spatial and non-spatial data separated, and geo-references are used as a means for visualization of the extracted patterns.

A tighter correlation between spatial and non-spatial data is given in [7], by observing that, in a spatial dataset, each object has an implicit neighborhood relation with the other objects in the set. The authors build a *neighborhood graph* where each node represents an object in the dataset and an edge between two nodes, say n_1 and n_2, represents a spatial relation between the objects associated with n_1 and n_2. The distance between two nodes in the graph (i.e. the number of edges in the path from the starting node to the target one) is used to weigh the properties associated to each node. This consideration gives the idea of *influence* between nodes, i.e. two nodes connected by a short path are likely to influence each other in the real world.

Some other papers explore the extension of knowledge discovery techniques to spatial domain. For example, in [11] an algorithm based on CLARANS for clustering spatial data is proposed.

In [9] and [10], two methods for extracting spatial association rules from a spatial dataset are presented.

Classification and ID3 Algorithm. A classification assigns a category to transactions according to the values of (some of) their attributes. Usually, the input is a table of tuples, where each row is a *transaction* and each column is an *attribute*. One of the attributes is chosen as the *class attribute*. The task is to build a model for predicting the value of the *class* attribute, knowing only the values of the others.

Many methods have been proposed for classification, such as neural networks, Bayesian classifiers and decision trees. We take into consideration decision tree models, since they proved to be very robust to noisy data, and in our context this property is crucial. The decision-tree classifiers proposed in the literature can be distinguished according to the statistical criterion used for splitting. For example CART [4] uses the *Gini index*, whereas *ID3* [13] and *C4.5* [14] use the *entropy* to measure (im)purity of samples. We focus our attention on ID3 algorithm and we show how it has been adapted to our purposes.

2 Spatial Data Model

Spatial data store the geometric description of geographical features, along with a state associated with these features. The state of a feature is a set of attributes. Many GIS tools organize spatial data as *vector* or *raster* maps. The correspondent attributes are stored in a set of tables related geographically to the geometric features [3, 6]. Digital maps are organized in *layers* (sometimes called *coverages* or *themes*). A layer represents a distinct set of geographic features in a particular domain. For example, a map of a country may have one layer representing cities, one representing roads, one representing rivers and lakes, and so on. Usually, a layer has several properties to control how it is rendered during visualization (i.e. the spatial extent, the range of scales at which the layers will be drawn on the screen, the colors for rendering features).

In [12], the Open GIS Consortium provides the specification for representing vector data to guarantee the interoperability and the portability of spatial data. We took advantage of the large availability of tools that support this specification (e.g. PostGIS [15], MySQL, GDAL libraries, JTS [8], GRASS GIS).

Given this organization of digital maps, it seems natural to maintain the same structure also during the knowledge discovery (and classification, in this case) task. The process of knowledge extraction aims at extracting implicit information from raw data. In a spatial dataset, the search for this implicit information is driven by the spatial relations among objects. For example in [7], a neighborhood graph is built to express these relationships. This graph is explored to extract useful information by finding, for example, similarities among paths.

Our point is to exploit the layer structure to select useful relations. For example, we may have interest to search "inter-layer" relations among objects (i.e. spatial relations where at least two members of the relation belong to distinct layers), rather than "intra-layer" relations. Thus, the heuristics proposed in [7] (i.e. following a path that tends to go away from its source) may be enriched by including only edges that represent "inter-layer" relations in the neighborhood graph. This assumption seems to be reasonable, since almost all papers on knowledge discovery present methods to extract patterns of this kind, i.e. patterns where the classes of objects involved belong to different layers. although the entity "layer" is not explicitly present.

We consider an input composed of n layers. Each layer L_i has only one geometric type: all the geometric features are polygons, or lines, or points. For the clarity of the presentation we consider for the moment only polygonal layers. Each layer has a set of attributes that describe the state of each object in the layer. We choose one of this attribute as the representative of the layer. Thus, each layer L_i represents a variable x_i over a domain X_i, where x_i is the chosen attribute. For example, a layer of the agricultural census data has several attributes associated with each polygon (e.g. the number of farms, the area of crop land, and other properties). If the attribute *number of farms* is chosen then each polygon in the layer can be considered as an instance of this variable.

Spatial Transactions. One of the layers is selected as the *class label* layer: we exploit these objects for selecting *spatial transaction*s. Each polygon represents an "area of interest". These areas are related to areas in the other layers in order to extract a set of tuples where each value in a tuple corresponds to the value in each of the layers, with respect to the intersection with the area of interest. So, like for relational databases, we now have a set of transactions. While relational transactions are "measured" by counting tuple occurrences in a table, we use here, as it is intuitive, the area extension of each spatial transaction.

Example We introduce here a simple example that will be useful for the whole presentation. Consider an agricultural environment, where information about crops is organized in layers as described above. For the sake of readability, in Figure 1 layers are presented side by side, even if they should be considered stacked one on top of the other. In Figure 1(f) the areas of samples are reported. Layers describe measures of qualities of the soil, like the availability of water (fig.1(a)) or potassium (fig.1(b)), or other parameters of the environment, like climate conditions (fig.1(c)). The training set, namely the *class label* layer (fig.1(d)), provides information about classified sample regions. In this example, we have a training set that represents crop types for some areas in the region.

3 Spatial Classification by Decision Tree Learning

Our goal is to build a decision tree capable of assigning an area to a class, given the values of the other layers with respect to the area. Like in transaction classification, we follow two steps: first, we build a model from a set of samples, namely a *training set*; then, we use the model to classify new (unseen) areas.

The training set is determined by the spatial transactions extracted from the dataset. We focus our attention on spatial relations derivable from *9-intersection model* [5] according to the adopted spatial model [12].

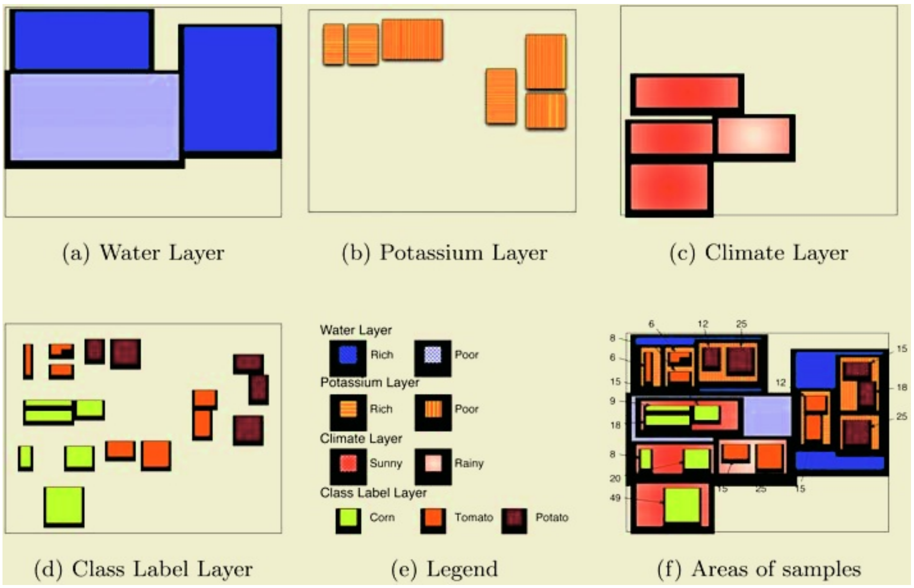

Fig. 1. An agricultural map

Spatial Decision Trees. A *Spatial Decision Tree* (SDT) is a rooted tree where *(i)* each internal node is a decision node over a layer, *(ii)* each branch denotes an outcome of the test and *(iii)* each leaf represents one of the class values.
A decision node n_i is associated with a layer L_i and with the attribute X_i of the layer. The outcoming edges are labeled with the possible values of X_i.

SDT Classification. An area A is classified by starting at the root node, testing the layer associated with this node and following the branch corresponding to the test result. Let $x_1, x_2, \ldots, x_m$ be the labels of the m edges of the root node. If A intersects an object of type x_j in the layer associated with the root node, then the edge labeled with x_j is followed. This testing process is repeated recursively starting from the selected child node until a leaf node is reached. The area A is classified according to the value in the leaf. When the query region A intersects several areas with distinct values, then all the corresponding branches are followed. The area A is split according to the layer values and each portion is classified independently.

Example In Figure 2(a) a spatial decision tree for the example in Figure 1 is presented. This decision tree classifies areas according to whether they are suitable for a type of crop rather than another. In particular, in this example we have three kind of crops: *Corn*, *Tomato* and *Potato*. Given a new instance **s** (marked with "??" in Figure 2(b)), we test **s** starting from the layer associated with the root node, i.e. the *Water* layer. Since **s** overlaps a water region whose value is *Poor*, the corresponding branch is followed and the node associated with

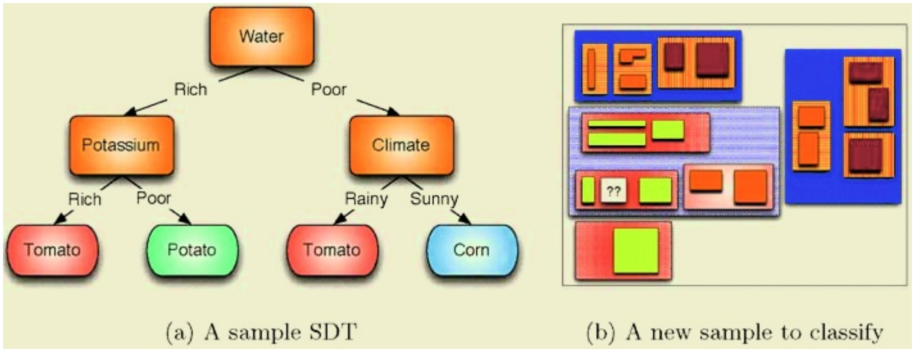

Fig. 2. A possible spatial decision tree

the *Climate* layer is selected. Thus, **s** is tested against the features in the *Climate* layer: in this case it overlaps a *Sunny* region, so the class *Tomato* is assigned to the instance **s**.

3.1 SDT Learning Algorithm

Following the basic decision tree learning algorithm [13], our method employs a top-down strategy to build the model. Initially, a layer is selected and associated with the root node, using a statistical test to verify how well it classifies all samples. Once a layer has been selected, a node is created and a branch is added for each possible value of the attribute of the layer. Then, the samples are distributed among the descendant nodes and the process is repeated for each subtree.

Algorithm 1 shows a general version of the learning method. Initially, termination conditions are tested to solve trivial configurations (for example, when all samples belong to the same class). The `majority_class(`S`)` is the class where most of the samples in S belong to. The crucial point of the algorithm is the selection of the split layer for the current node. In Section 3.3 a strategy based on the notion of entropy is presented to quantify how well a layer separates samples. Once a layer is selected for a test node, the samples are partitioned according to the layer itself and the intersection spatial relation. In Section 3.2 we show how to compute this partition.

3.2 Splitting Samples

When classifying transactions represented as tuples, we aim at grouping transactions together according to an attribute A. If the attribute for splitting is selected in a proper way, the samples in each subpartition may increase their uniformity (or, in other terms, they reduce their entropy).

In the same way, we aim at grouping spatial samples according to the information found in the other layers. We select a layer L_i and we split the samples in

Algorithm 1: Generate_SDT

```
Input: A layer $S$ of sample areas;
       A list $\mathcal{L}$ of layers;
Output: A spatial decision tree
Create a new node $N$;
if samples in $S$ are all of class $c$ then
    label $N$ with $c$;
    exit;
end
if $\mathcal{L}$ is empty then
    label $N$ with majority_class($S$);
    exit;
end
Select layer best_split from $\mathcal{L}$;
Split $S$ according to layer best_split in $\{S(c_1), \ldots, S(c_p)\}$;
foreach $S(c_i)$, $i = 1, 2, \ldots, p$ do
    Let $N_i$ = Generate_SDT($S(c_i)$, $\mathcal{L} \setminus \{$best_split$\}$);
    Create a branch from $N$ to $N_i$ labeled with the selected value;
end
```

layer S according to this layer. In general, if layer L_i has q possible values then it can split the samples in $q+1$ subsets, i.e. a subset for each value v_j,$j = 1, 2, \ldots, q$, and a special subset correspondig to none of these values (termed $\neg L(C)$). We use $L_i(v_j)$ to refer to all the features in L_i that have value v_j.

The *intersection* relation gives us the possibility to express a quantitative measure on the related objects. In fact, given two polygonal geometries, say g_1 and g_2, we obtain a new geometry by means of the function *intersection*(g_1, g_2). If the two geometries overlap and their intersection geometry is g_3 then the *area* of g_3 gives the quantitative measure of the relation (under the assumption that g_3 is a polygon; the other cases are discussed below).

Given the subset $L_i(v_j)$, for $j = 1, 2, \ldots, q$, we consider all the samples that intersect any feature in $L_i(v_j)$. We denote this sublayer as $L_i(v_j, C)$. For each class value c_k, we denote with $L_i(v_j, c_k)$ the features in $L_i(v_j, C)$ whose class is c_k. Clearly, when a sample overlaps partially a polygon with value v_k and a polygon with value v_l (this situtation is showed in Figure 3) it is split: first, a portion of the sample is computed by intersecting it with polygon v_k; the remaining part of the sample is related to the other polygon; the possibly remaining portion of the sample is left unclassified. For example, the sample in Figure 3(a) is partitioned into three samples (Figure 3(b)).

For the sake of simplicity, we may think of each feature in $L_i(v_j, c_k)$ as a representative for a tuple (v_j, c_k). While in a tuple-transactions context we use cardinality as a quantitative measure, we adopt the *area* as a quantitative measure for “spatial tuples”.

Example In Figure 3(c) the result of splitting the samples according to *Water* Layer is showed. There is just one feature in the layer $L_{Water}(Poor)$. Then all the

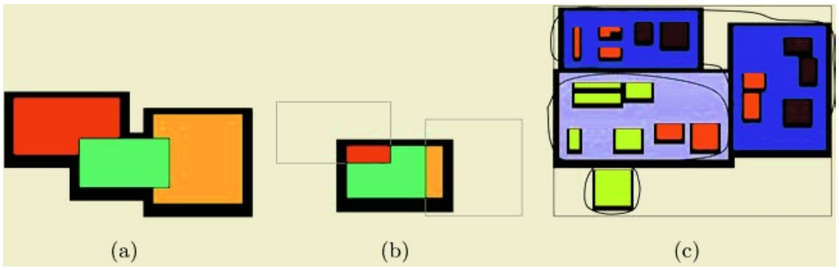

Fig. 3. (a) A sample (at center) overlaps two polygons; (b) the sample after splitting; (c) samples splitted according to the *Water* layer

samples that intersect this object belong to the same subset. The two polygons in $L_{Water}(Rich)$ enclose the other subset of samples. We denote, respectively, with $L_{Water}(Poor, C)$ and $L_{Water}(Rich, C)$ the two subsets of samples. In particular, $L_{Water}(Poor, C)$ is the union of two layers: $L_{Water}(Poor, Corn)$, that contains the five corn polygons on the left, and $L_{Water}(Poor, Tomato)$, that contains the other two polygons.

One of the samples does not overlap any feature either in $L_{Water}(Poor)$ nor in $L_{Water}(Rich)$. Thus, the sample is inserted in $\neg L_{Water}(C)$.

3.3 Selecting Best Split

At each step of the algorithm, we choose one of the candidate layers for growing the tree and for separating the samples. In this section we introduce a statistical measure, the *spatial information gain*, to select a layer that classifies training samples better than the others. The information gain is based on the notion of *entropy*. Intuitively, it is possible to measure the (im)purity of samples w.r.t. their classes by an evaluation of their entropy. Then, candidates for splitting are selected considering the reduction of entropy caused by splitting the samples according to each layer.

Spatial Information Gain. We present now the method to compute the entropy for a layer L, with respect to the class label layer S. First, we evaluate the entropy of the samples, i.e. the information needed to identify the class of a spatial transaction. While in tuple-transaction the frequency of a sample is expressed as a ratio of transaction occurences, we use here the area extent of the samples.

Thus, given a layer L, we denote with $mes(L)$ the sum of the areas of all polygons in L. If S has l distinct classes (i.e. $c_1, c_2, \ldots, c_l$) then the entropy for S is:

$$H(S) = -\sum_{i=1}^{l} \frac{mes(S_{c_i})}{mes(S)} log_2 \frac{mes(S_{c_i})}{mes(S)} \tag{1}$$

If L is a non-class layer with values $v_1, v_2, \ldots, v_q$, we split the samples according to this layer, as showed in Section 3.2. We obtain a set of layers $L(v_i, S)$ for each possible value v_i in L and, possibly, $\neg L(S)$. From equation (1) we can compute the entropy for samples in each sublayer $L(v_i, S)$. The expected entropy value for splitting is given by:

$$H(S|L) = \frac{mes(\neg L(S))}{mes(S)} H(\neg L(S)) + \sum_{j=1}^{q} \frac{mes(L(v_j, S))}{mes(S)} H(L(v_j, S)) \tag{2}$$

The layer $\neg L(S)$ represents the samples that can not be classified by the layer L (i.e. the samples not intersected by the layer L). This scenario may happen, for example, when layer L partially covers the *class label* layer. Thus, when selecting layer L for splitting, we consider the entropy of the samples with empty intersection in the computation of the expected entropy. While the values of layer L may be used to label the edges of the tree, the values of layer $\neg L(S)$ are used as an "handicap" for the layer entropy. For example, consider a layer L containing a single polygon that covers only a small portion of the *class label* layer. The splitting will produce a layer $L(v, S)$ (corresponding to the unique value of layer L) and a layer $\neg L(S)$ larger than the first one. By considering only the layer $L(v, S)$ the resulting entropy would be very low, but a larger part of the sample would remain unclassified. Instead, the entropy measure of layer $\neg L(S)$ gives a measure of the "remaining impurity" left to classify.
The *spatial information gain* for layer L is given by:

$$\text{Gain}(L) = H(S) - H(S|L) \tag{3}$$

Clearly, the layer L that presents the highest gain is chosen as *best split*: we create a node associated with L and an edge for each value of the layer. The samples are splitted among the edges according to each edge value. The selection process is repeated for each branch of the node by considering all the layers except L.

Example Consider the splitting of samples in Figure 1(d) with respect to the *Potassium* layer (Figure 1(b)). The entropy of layers $L_{\text{Potas}}(\text{Rich})$ and L_{Potas} (Poor) is zero. By ignoring the $\neg L_{\text{Potas}}(S)$ the overall entropy of splitting would be zero. Thus, the *Potassium* layer would be selected for splitting. However, so doing, the information of samples in $\neg L_{\text{Potas}}(S)$ is lost: in fact, the $\neg L_{(\cdot)}(S)$ is not used to build the tree.

To clarify the layer selection task, consider the training set in Figure 1(d). Here, the *class label* layer contains polygons whose classes represent crops relative to each area. Following Algorithm 1, we start by building the root node for the tree. One of the available layers has to be selected. So the information gain is computed for each one (i.e. *Water*, *Potassium*, *Climate*).

For instance, we show the information gain computation for the *Water* layer. The splitting of samples according to the *Water* layer is reported in Section 3.2 and it results into three sublayers: $L_{Water}(Poor, C)$, $L_{Water}(Rich, C)$ and

$\neg L_{Water}(C)$. The areas of the samples are reported in Figure 1(f). The entropy of each layer is given by, respectively:

$$H(L_{Water}(Poor,C)) = -\frac{70}{110}log_2\frac{70}{110} - \frac{40}{110}log_2\frac{40}{110} - \frac{0}{110}log_2\frac{0}{110} = 0.9457$$
$$H(L_{Water}(Rich,C)) = -\frac{0}{142}log_2\frac{0}{142} - \frac{47}{142}log_2\frac{47}{142} - \frac{95}{142}log_2\frac{95}{142} = 0.9159$$
$$H(\neg L_{Water}(C)) = -\frac{49}{49}log_2\frac{49}{49} - \frac{0}{49}log_2\frac{0}{49} - \frac{0}{49}log_2\frac{0}{49} = 0.0000$$

From (2), the expected entropy for splitting is:

$$H(S|L_{Water}) = -\frac{110}{301} \times 0.9457 - \frac{142}{301} \times 0.9159 - \frac{49}{301} \times 0.0000 = 0.1830$$

Since the entropy is:

$$H(S) = -\frac{119}{301}log_2\frac{119}{301} - \frac{87}{301}log_2\frac{87}{301} - \frac{95}{301}log_2\frac{95}{301} = 1.5718$$

from (3) we can compute the gain for the *Water* layer (and for the other two layers):

$$Gain(L_{Water}) = 1.3888;\ Gain(L_{Potassium}) = 1.2331;\ Gain(L_{Climate}) = 1.2742;$$

Since the *Water* layer shows the best gain it is selected as root test layer.

3.4 Experiments

We have performed some experiments with real digital maps. In particular, we considered a set of layers from the National Atlas of the United States:

- the *class label* layer contains the information about the average (per farm) market value of sold agricultural products;
- the *ecoregion* layer represents areas that present a common climate;
- the *aquifr* layer shows the distribution of the principal aquifers.
- *cotton*, *soybean*, *wheat* layers specify the areas cultivated with cotton, soybeans, wheat for grain respectively;
- *cattle* layers gives the number of cattle and calves per area.

All the layers, but *ecoregion* and *aquifr*, have continuous attributes. We have discretized each attribute by grouping objects into classes with comparable areas.

The spatial operations and indexing are handled by means of the *JTS Topology Suite*[8]. Each layer is indexed with a STRTree [16]. This reduces drastically the execution time for the splitting operation. At each split operation:

- for each polygon in the *class label* layer we compute the intersection with the current layer. The operation is performed in two steps: first, a coarse query is executed on the spatial index; then, the query response is refined and the result is inserted into a new layer;

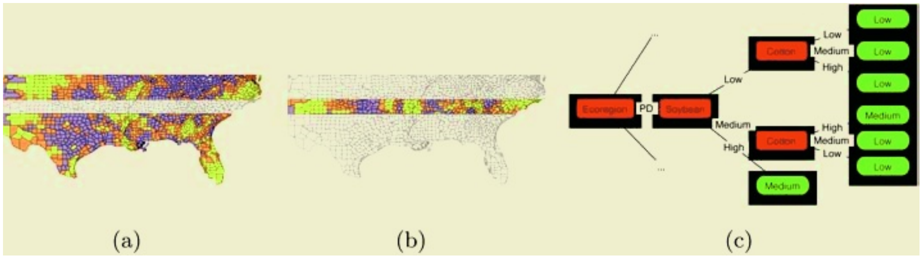

Fig. 4. (a) A portion of a geographical map; (b) the test set considered; (c) the SDT extracted

– the new layer L is partitioned into sublayers $L(v_i)$, for each possible value v_i, to compute the entropy. To speed up this operation, we create an additional hashtable-based index to retrieve each of the sublayer by class.

One of the datasets used for the experiments is showed in Figure 4(a) (the corresponding test set is reported in Figure 4(b)). Continuous layers have been classified into three classes, namely *Low*, *Medium* and *High*. The extracted SDT (Figure 4(c) presents a branch of the whole tree) has the root node associated with the *ecoregion* layer. The accuracy reached on the test set is about 80%: the whole area of the test set is 48.1217; the area of correctly classified polygons is about 39.1.

4 Conclusions and Future Work

We have presented a method for learning a classifier for geographical areas. The method is based on the ID3 learning algorithm and the entropy computation is based on the area of polygons. However, the requirement for polygonal layers may be relaxed and we are currently investigating several possible solutions. It is possible to use line layers (for example, road layer) as well. In this case, the measure of the intersection is computed by considering the *length* of linear geometries. For point layers a suitable measure is the *count* of the occurrences of points. In practice, the entropy measure would be biased to polygonal layers. A good tradeoff is to consider a buffered layer (where each polygon contains all the points within a given distance to an object in the original layer). This solution creates a fresh polygonal layer. However, the choice of the buffer size is not so simple: it can also produce a bias as in the previous scenario. Another solution is to consider the definition of an entropy function *weighted* on each layer. This solution may provide the user with the possibility of promoting some layers during the learning process (i.e. for specifying the interest toward a specific layer). We are also considering some heuristics to improve the quality of data preparation, in particular the discretization of layers with numeric attributes. Another possible direction of research is considering the problem of learning spatially explicit classification models based on intra-layer relations, i.e. topological relations among neighboring spatial objects.

References

1. G. L. Andrienko and N. V. Andrienko. Data mining with c4.5 and cartographic visualization. In N.W.Paton and T.Griffiths, editors, *User Interfaces to Data Intensive Systems*, IEEE Computer Society Los Alamitos, CA, 1999.
2. G. L. Andrienko and N. V. Andrienko. Knowledge-based visualization to support spatial data mining. In D. J. Hand, J. N. Kok, and M. R. Berthold, editors, *Third Int. Symp. on Advances in Intelligent Data Analysis, IDA-99*, volume 1642. Springer, 1999.
3. R. Blazek, M. Neteler, and R. Micarelli. The new GRASS 5.1 vector architecture. In *Open source GIS – GRASS users conference 2002, Trento, Italy, 11-13 September 2002*. University of Trento, Italy, 2002.
4. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.
5. M. Egenhofer. Reasoning about binary topological relations. In A. P. Buchmann, O. Günther, T. R. Smith, and Y. F. Wang, editors, *Proc. of the 2nd Int. Symp. on Large Spatial Databases (SSD)*, LNCS. Springer-Verlag, 1989.
6. ESRI - Arcview. Available at http://www.esri.com/software/arcview/.
7. M. Ester, A. Frommelt, H. Kriegel, and J. Sander. Spatial data mining: Database primitives, algorithms and efficient DBMS support. *Data Mining and Knowledge Discovery*, 4(2/3):193–216, 2000.
8. Vivid Solutions Inc. - JTS Topology Suite v1.4, 2000.
9. K. Koperski and J. W. Han. Discovery of spatial association rules in geographic information databases. *LNCS*, 951:47–66, 1995.
10. D. Malerba, F. Esposito, and F.A. Lisi. Mining spatial association rules in census data. In *Proc. of the Joint Conf. on "New Techniques and Technologies for Statistcs" and "Exchange of Technology and Know-how"*, 2001.
11. R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proc. of the 20th Int. Conf. on Very Large Databases*, 1994.
12. Simple feature specification. Available at http://www.opengis.org/specs, 1999.
13. J R Quinlan. Induction of decision trees. *Machine Learning*, 1(1), 1986. QUINLAN86.
14. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.
15. Postgis: Geographic objects for PostgreSQL.
    Available at http://postgis.refractions.net, 2002.
16. Philippe Rigaux, Michel Scholl, and Agnes Voisard. *Spatial Databases: With Application to GIS*. Morgan Kaufmann, 2002.