A Multi-Relational Approach to Spatial Classification

Richard Frank
School of Computing Science
Simon Fraser University
Burnaby BC, Canada
rfrank@cs.sfu.ca

Martin Ester
School of Computing Science
Simon Fraser University
Burnaby BC, Canada
ester@cs.sfu.ca

Arno Knobbe
LIACS, Leiden University
Leiden, the Netherlands
knobbe@liacs.nl

ABSTRACT

Spatial classification is the task of learning models to predict class labels based on the features of entities as well as the spatial relationships to other entities and their features. Spatial data can be represented as multi-relational data, however it presents novel challenges not present in multi-relational problems. One such problem is that spatial relationships are embedded in space, unknown a priori, and it is part of the algorithm's task to determine which relationships are important and what properties to consider. In order to determine when two entities are spatially related in an adaptive and non-parametric way, we propose a Voronoi-based neighbourhood definition upon which spatial literals can be built. Properties of these neighbourhoods also need to be described and used for classification purposes. Non-spatial aggregation literals already exist within the multi-relational framework, but are not sufficient for comprehensive spatial classification. A formal set of additions to the multi-relational data mining framework is proposed, to be able to represent spatial aggregations as well as spatial features and literals. These additions allow for capturing more complex interactions and spatial occurrences such as spatial trends. In order to more efficiently perform the rule learning and exploit powerful multi-processor machines, a scalable parallelized method capable of reducing the runtime by several factors is presented. The method is compared against existing methods by experimental evaluation on a real world crime dataset which demonstrate the importance of the neighbourhood definition and the advantages of parallelization.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – *Data mining, spatial databases and GIS*

General Terms

Algorithms, Experimentation, Theory

Keywords

spatial data mining, spatial classification, aggregation, parallelization

1. INTRODUCTION

This paper is concerned with Data Mining in spatial data. Specifically, we are interested in the discovery of predictive patterns that help solve a spatial classification task [4]. For example, we may be interested why a house will be burgled, in terms of its spatial characteristics, such as location, the nature of neighbouring houses and the proximity of other spatial entities (roads, shops, etc.). Due to the relational nature of the data, it is logical to approach the spatial classification task as a multi-relational one. We will show that additional techniques are required to deal with issues that are unique to the spatial domain.

Although multi-relational (MR) techniques are a promising start, they cannot be applied directly to spatial data since MR and spatial data are different in the way relationships between entities are defined. In an MR database, the relationships are explicitly given but with spatial data, these same relationships are only implied through the spatial location of the entities themselves. Therefore, in order to involve information regarding neighbouring entities, these relationships need to be made more explicit.

As an additional complication of spatial data, relationships in spatial data can be numerous: for a typical mall there are literally thousands of houses scattered near it. Hence, each individual relationship can become insignificant on its own, requiring the use of some form of (spatial) aggregation. Simple aggregation however is not enough. In the multi-relational domain, dependencies between features of entities might exist, but for spatial entities, they are known to play a crucial role. This dependency is based on *Tobler's First Law of Geography* [20] which states that "the larger the distance between entities, the more negligible their effects on each other".

The field of Multi-Relational Data Mining (MRDM) [22] deals with data organized into types with each type t having a (possibly) different set of features. Entities in t are related to entities of other types, or to other entities in t itself. The classification process starts at a specific type, called the target type τ. All entities in τ have exactly one of multiple possible class labels assigned to them. The model learning involves features of the target type τ, and expands out to other types in order to involve additional information related to τ. Similar to a multi-relational database, a spatial database S also contains a set of entity types with t denoting a specific type, for example *house* or *mall*. Each entity has features being of one type from the set {date, numeric, categorical, spatial}. The goal of spatial classification is to find patterns and learn predictive models based on interactions of spatial and non-spatial features of the entities in S.

In this paper, we explore the multi-relational approach to spatial classification. A novel comprehensive spatial classification rule-learner is proposed, called ***Un**ified **M**ulti-relational **A**ggregation-*

*based **S**patial **C**lassifier* (UnMASC). It builds neighbourhood relationships via a novel Voronoi-based approach, with rules incorporating a broad range of spatial and aggregation literals that are created on-the-fly. As our running example, we will use the spatial database shown in Figure 1, which contains information about three spatial and one non-spatial entity-types. Note the lack of (non-spatial) relationships between the spatial entities. Each spatial entity has a shape feature that contains the polygonal representation of the entity, from which the area, perimeter, location and other spatial features can be derived.

The contributions of this paper are as follows:

- Formulating the spatial classification problem as a MR one. This gives the rule-learner a solid theoretical foundation and allows for new insights into the problem and solution
- Uniquely establishing neighbourhood relationships between entities via a novel non-parametric Voronoi-based approach which filters out irrelevant relationships
- Extending the MRDM framework to represent spatial aggregation using functions and literals
- Presenting a parallel implementation of the above techniques through UnMASC
- Experimentally evaluating UnMASC on real-world data

In Section 2 the Voronoi neighbourhood definition is contrasted to existing methods. Section 3 presents the formal MRDM framework and the extensions required for spatial data mining. The algorithm is discussed in Section 4, while experimental results are shown in Section 5.

2. NEIGHBOURHOOD DEFINITIONS

Since spatial data only indirectly implies relationships via the spatial location of the entities, some work has to be done to explicitly determine whether two entities are related. Tobler's First Law of Geography implies that relationships are important to different degrees, and this importance is influenced by distance. This means that a spatial classifier needs to focus on neighboring entities in order to save analyzing unlikely dependencies between distant entities and avoid constructing misleading rules. In the following, we review state-of-the-art neighborhood definitions from the literature, *topological relationships* [4, 15, 17] and *buffer zones* [6, 2], and argue that they are inadequate for the purposes of spatial classification. Finally, we introduce a novel neighborhood definition based on Voronoi diagrams.

Topological relationships, preserved under translation, rotation or scaling [7], define the relationship between two entities in terms of eight predicates that describe the intersections of their boundary, interior and exterior [15] (e.g. *meet*, *overlap*). They are very common and easy to understand, but do not always capture the intuitive notion of a neighbourhood. Many people would define themselves as living in the neighbourhood of a mall even if they lived in the close vicinity and not directly adjacent to it.

Buffer zones are more commonly used for finding the neighbourhood of an entity. The buffer zone of size d around an entity i of type t, denoted $e^{t,i}$, is the area that is within distance d to $e^{t,i}$ (Figure 2). Any entities intersecting the buffer zone are defined to be neighbours of $e^{t,i}$. Unfortunately, buffer zones do have major drawbacks. First, for entity types of greatly varying size and range of influence, a constant sized buffer zone (3a) either becomes too large (3b) or becomes irrelevant (3c). Second, in principle, an infinite number of buffer zone sizes could be selected, and each size possibly changes the resulting rules [18]. Third, for all entities in t, the distribution can change significantly across the entire dataset (Figure 4).

What is needed is a neighbourhood definition that can take into account the number of entities and their distribution, ideally one without user input. The notion we adopt here is that of the *Voronoi diagram*. Voronoi diagrams partition a plane into regions, called Voronoi cells, which contains the area that is closest to the entity contained in the Voronoi cell, naturally representing relationships between entities [3]. Voronoi diagrams can be computed for point data (e.g. houses), segment data (e.g. roads), and areal data (e.g. lakes) [10]. Note that we use all these data-types in our experiments (see Section 5).

Voronoi diagrams have been used in the domain of computer geometry [10]. With the aid of Voronoi diagrams, entities of a *single* type were used for clustering [8], and outlier detection [1]. As opposed to having only a single entity-type, [3] used Voronoi diagrams for finding spatial association rules for *multiple* entity-types. However, though the dataset contained multiple entity-types, the Voronoi diagram itself was created as if all entity-types were the same. This does not take advantage of the fact that the entity-types are of different characteristics, and creates a neighbourhood structure that connects adjacent entities only.

A definition is required that creates meaningful neighbourhood relationships between multiple types of entities in spatial data, while preserving the importance of different types. People tend to go to the closest mall, food-store, hospital or airport, and the definition needs to take advantage of this. Based on this idea, for each entity-type, we construct a different neighbourhood structure using Voronoi diagrams. This is defined as follows:

DEFINITION 1. **(Voronoi Neighbourhood)** *Two entities,* $e^{t,q}$ *and* $e^{s,r}$*, are neighbours iff:*

- $e^{t,q}$ *intersects the Voronoi cell of* $e^{s,r}$ *or* $e^{s,r}$ *intersects the Voronoi cell of* $e^{t,q}$*, and* $e^{t,q}$ *and* $e^{s,r}$ *are different entity types, i.e.*: $t \neq s$, *or*,
- *the Voronoi cells of* $e^{t,q}$ *and* $e^{s,r}$ *are adjacent, and* $e^{t,q}$ *and* $e^{s,r}$ *are of the same type, i.e.*: $t = s$.

This implies that entities close to $e^{t,q}$ have the most influence on $e^{t,q}$, which is consistent with Tobler's First Law of Geography. The proposed approach cannot, and purposely does not, find

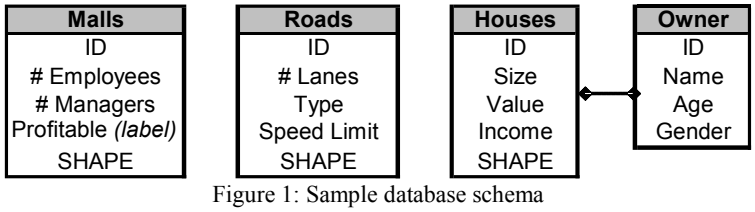


Figure 1: Sample database schema

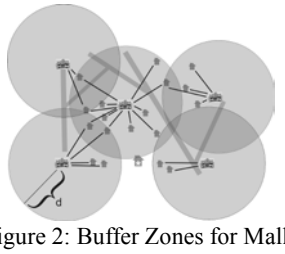


Figure 2: Buffer Zones for Malls

patterns at arbitrary proximities since it makes the assumption that patterns that do not appear in close proximity (that is, connected by a *neighbour* literal) are not interesting. The Voronoi partitioning is completely data-driven and does not rely on any domain knowledge of the user. Thus the proximity for each entity type is derived from the distribution and number of entities of that entity type. For regions with lots of houses, for example, the proximity will be small; however for entity types with only few entities, such as airports or hospitals, the proximity will be very large. This creates a very natural concept of neighbourhood as, for example, each house is a neighbour to their closest malls, and each mall is a neighbour to other nearby malls (Figure 5).

3. MR RULES FOR SPATIAL MINING

To learn classification rules on a spatial dataset S, a target entity type τ and a class label from τ are selected by the user. Each entity in τ, called a target entity, has exactly one class label assigned. The goal of classification is to find rules that predict which class a target entity belongs to, given its own location, feature values, relationships to other entities, their locations and features. The entities of τ can either be interrelated to other entities also in τ or related to entities of other types. In instances when they are interrelated, the class label may depend on other entities of the same type and their class labels [19].

3.1 Preliminaries

As customary in Multi-Relational Data Mining, we will use first order logic (FOL) clauses [5] to represent classification rules. In FOL, entities are represented by *constants*, and comparisons of terms to constants are expressed by *literals,* which make up *rules*:

DEFINITION 2. **Terms** *are constants and variables.*

DEFINITION 3. *A* **literal** *is a mapping of terms to a Boolean value, or a comparison* θ *of a term to a constant, where* $\theta \in \{=,<,\leq,>,\geq\}$.

DEFINITION 4. *A* **classification rule** *is of the form* $L_0 \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_n$. *Where each* L_i *is a literal, and* L_0 *specifies a class-label.*

The literals are implicitly existentially quantified. As an example:

R_1: profitable(M, 'yes') ← mall(M), neighbour(M, H), house(H), income(H, I), I > 100,000

states that "*a mall is profitable if there exists a neighbouring house with income greater than* $100,000". Here house *H* is implicitly existentially quantified. *mall(M)* is derived from entity table *mall*, *neighbour* denotes a relationship between *mall* and *house*, and *income* is a feature of *house*. The rule requires at least one house, but it is likely that most malls have at least one neighbouring house with income greater than $100,000. Therefore, R_1 is relatively weak. What could help is determining how many such houses are neighbours.

With MR and spatial data, any single entity in t_i can be related to multiple other entities in t_j. To describe the feature values of these related entities, *aggregation functions* can be used. Chelghoum et al. [6] present a pre-processing technique in which a separate table is created to summarize the relationships between t_i and t_j. The drawback is that as the rule is being built, a condition could be placed on some feature of t_i (for example *size*(*H, S*), *S* = *'large'*), which would invalidate the pre-aggregated values. Furthermore, there are a large number of aggregations, which typically yields a table of considerable size [11, 21]. The aggregations presented in this paper could in theory also be applied as a pre-processing step, but then would suffer from these same limitations. In order not to be limited, features must be aggregated during the literal search and not a priori.

3.2 Single and Multi-Feature Aggregation

Recently, FOL has been extended to include aggregation using functions involving single [11, 21] and multiple [9] features:

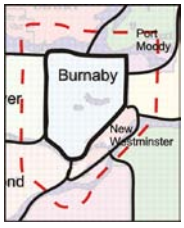
DEFINITION 5 *A* **single-feature aggregation (SFA) function** *maps a bag of elements from the domain of a feature to a single value from another (possibly different) domain.*

DEFINITION 6 *A* **single-feature aggregation literal** *has the form* Agg(input, {conditions}, result) *where* input *is a variable specifying the bag of feature values to be aggregated by* Agg*, constrained by* conditions*, and* result *is an output variable referencing the result of the aggregation.*

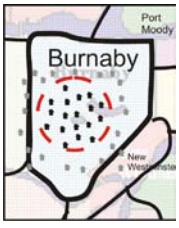
DEFINITION 7. *A* **multi-feature aggregation (MFA) function** *maps multiple lists of elements from the domains of the features to a single value of another domain.*

DEFINITION 8. *A* **multi-feature aggregation literal** *has the form* Agg({$input_1$, $input_2$, …, $input_i$}, {conditions}, result) *where* input *specifies lists of corresponding feature values aggregated and constrained by* conditions*.* result *is the output variable of the MFA function corresponding to* Agg.

Single-feature aggregations include functions such as *count*. Multiple features can also create literals by applying functions like *correlation* and *t-test*. The exact functions available depend on the type of feature(s) being aggregated. A comprehensive list of aggregations used in the literature is presented in [9].



a) around a city



b) around a house

c) around a province

Figure 3: 3 km buffer zone around different entity types

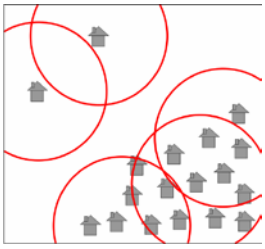


Figure 4: Varying density buffer zones

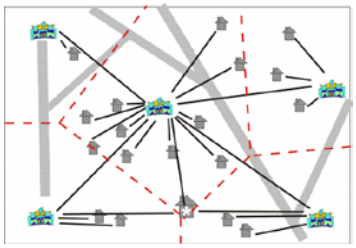


Figure 5: Voronoi neighbourhood definition (shadings represent roads)

As an example, using these constructs, we can capture *average value* and *correlation* between *income* and *size* of houses yielding

R_2: profitable(M, 'yes') ← mall(M),
AVG(V,{neighbour(M, H), house(H), value(H, V)}, A),
CORR({I, S}, {neighbour(M, H), house(H),
income(H, I), size(H, S)}, C), A > 100,000, C < 0

With R_2 it is possible to express that *"a mall is profitable if, for neighbouring houses, the average value is greater than $100,000 while the house-hold income and size of house are negatively correlated"*. This rule implies that higher income earners live in smaller houses, leading to a possible higher disposable income.

3.3 Spatial Features

Spatial data is unique from non-spatial data in that it has implicit features, such as location or size, to be derived from the polygon associated with the entity. Some simple spatial features (Figure 6) have been used in the literature [2, 17], but are not appropriate in describing entities like road-segments or parks (the *perimeter* of 'Highway 1' is not useful). To increase the expressiveness of the language and measure interesting properties of 2D entities, a comprehensive and appropriate *independent* set of features needs to be introduced (Figure 7) using spatial features:

DEFINITION 9. *A* **spatial function***, with at least 1 parameter, returns a calculated property for the input(s).*

DEFINITION 10. *A* **spatial feature** *is a spatial function applied to a spatial variable. It describes some property of the input entity and has the form* SpatFeat(input)*. A condition and a threshold could also be applied.*

Using this construct, different spatial functions from any domain can easily be incorporated into the multi-relational rule-learning process. For example, *catchment_area*(*M*) denotes the area of the Voronoi cell that mall *M* is in, motivated by research in marketing [14]. This is defined as the area and population from which some entity (*mall* for example) attracts customers. Since spatial data, by default, does not contain explicit relationships, these need to be expressed and are added to FOL as *spatial literals*:

DEFINITION 11. *A* **spatial literal** *represents the relationship between two spatial entities. It has the form* SpatLit($input_1$, $input_2$) *or* SpatLit($input_1$, $input_2$) θ v*, depending on whether a comparison and threshold are applicable and used.*

DEFINITION 12. *A* **neighbour literal** *is a special spatial literal representing the existence of a relationship between two spatial entities, as defined by a neighbourhood definition. It has the form* neighbour($input_1$, $input_2$).

For example, *house*(*H*) and *mall*(*M*) can be connected with spatial literal *neighbour*(*H, M*). The literal *neighbour(H,M)* was created because, according to the neighbourhood definition, such as the buffer-zone or Voronoi neighbourhood, there is a relationship between *H* and *M*. Some spatial literals have already been explored [7], but a few critical additional spatial literals (Figure 8) need to be added to take advantage of the extra information available due to the use of Voronoi diagrams.

3.4 Spatial Aggregation

Since, in general, spatial data contains a lot of entities belonging to the same type, the effect of a single entity becomes relatively limited. Aggregation-based methods [21, 11, 9] address this problem, as described in the previous sections. However, applying existing (non-spatial) methods to spatial data might yield unreliable results. As an example, regression analysis does not adjust for spatial dependency and thus can have unreliable parameter estimates and significance tests [13]. Analogous to the format of multi-feature aggregation literals, the following *spatial aggregation literals* are added to the language:

DEFINITION 13. *A* **spatial aggregation (SA) literal** *has the form* Agg({$input_1$, $input_2$, …, $input_i$}, {conditions}, result) *where* $i \geq 1$ *and* input *specifies the lists of corresponding feature values aggregated and constrained by* conditions*.* result *references the result of the aggregate function corresponding to* Agg.

Figure 9 presents three spatial aggregation literals incorporated in our framework. They represent the spatial statistics functions most commonly used in the literature [12]. Spatial trends (e.g. *trend*) describe the correlation of a (non-)spatial feature-value *f* and the distance *d* away from a central entity. If *d* were replaced with another, non-spatial, feature, this measure is equivalent to *correlation*. Spatial trend has a domain of [-1,1] with the value denoting the direction of linear relationship (for example, -1 meaning *f* decreases as *d* increases).

Spatial autocorrelation (*autocorrelation*) measures changes in *f* with respect to the values of *f* of the neighbouring entities. Since the non-spatial aggregate function SUM can be misleading due to the effect of the different sizes of the neighbourhoods, *area_adjusted_mean* yields a spatially-averaged value. For example, pollution level for a factory is not an absolute number but a *per* km^2 value, i.e. it has been adjusted for the area it covers. As an example, using the literal *trend*() and a condition "*result* < 0", the rule can now capture a relationship of decreasing income and increasing distance relative to a central mall:

R_4: profitable(M, 'yes') ← mall(M),
TREND({I, D}, {neighbour(M, H), house(H),
income(H, I), distance(M, H, D)}, S), S < 0

This rule can be used to maximize the profit of a mall by advertising more to the close, high income, houses. Further, investors considering locations for future malls can place malls close to high income neighbourhoods.

Spatial Features				Spatial Literals	Spatial Aggregation Literals
- length	- width	- start_y	- end_x	- voronoi_neighbour	- trend
- perimeter		- end_y	- start_x	- road_distance	- autocorrelation
- area		- centroid_x	- centroid_y	- travel_time	- area_adjusted_mean
- x	- y	- catchment_area			

Figure 6: Existing spatial features

Figure 7: Novel spatial features

Figure 8: Novel spatial literals

Figure 9: Spatial Aggregation Literals

4. RULE LEARNING

Our MR-based spatial classification algorithm, UnMASC (**Un**ified **M**ultifeature **A**ggregation based **S**patial **C**lassifier), is based on the idea of the sequential covering algorithm [22]. The goal is to learn classification rules on entities with known class-labels (training entities), then use those rules to predict the label for new entities (testing entities). The learning and prediction is based on the entity's own feature values, (spatial) relationships to other entities and their feature values. The specific relationships materialized are given by the neighbourhood definition (such as buffer-zone or Voronoi neighbourhood) which is specified a priori. Spatial calculations are performed in the DBMS. We focus on the two-class classification problem.

In current algorithms, such as [22], the search space is explored serially as each evaluation takes place one after the other (Section 4.1). With multi-processor machines available today, this method of evaluation is inefficient since it is unable to exploit the available processing power. Our algorithm (Section 4.2) is able to perform this search process in an optimized parallelized fashion. We describe this parallelization, and the optimizations done to further improve the benefits of parallelization (Section 4.3).

4.1 Preliminaries

The learning of rules is done by generating one rule at a time and refining them incrementally by adding literals until a termination condition applies (for example Minimum Support). Once a rule is finalised, the covered entities are removed from the training set, and a search for another rule starts. The rule-learning process ends when there are not enough entities in the training set for a rule.

When refining a rule, the rule is extended by at least one literal at a time. If the entity type t in the literal being added is already referenced in the rule, then a feature literal is added with a new condition on some feature(s) of t. If the entity type in the literal has not been referenced in the rule, then up to three literals are added: an Entity Literal referencing t, a Relationship Literal denoting the relationship of t to an entity type already in the rule, and possibly a Feature Literal with a condition on a feature of t.

Each new rule references the target entity type τ. When searching for the next best literal, each entity type t that shares a (spatial) relationship with τ is searched. The search finds the feature, aggregation, constant value and comparison operator that yields the highest FOILGain [16], which is then added to the rule as a new literal. Note that depending on the neighbourhood definition, entities of type t could be neighbours to other entities of type t. For non-empty rules the search is more complicated since any entity type with a relationship to another entity type referenced by the rule must be evaluated. The more entity types referenced, the more relationships there are, and the larger the number of candidate literals that need to be evaluated.

4.2 The UnMASC Algorithm

When evaluating refinements of the current rule, the entity types that are considered include each entity type (*neigEntity*) that neighbours currently referenced entity types. The set of *neigEntity*'s is called *neigEntitySet*. For each *neigEntity*, the FOILGain of all aggregations applied to their corresponding feature(s) is evaluated independently of the other evaluations. The decision of which literal to add to the rule is being made only after the FOILGain has been evaluated for all *neigEntity*'s. This indicates that the evaluation of multiple *neigEntity*'s can be performed in parallel. UnMASC takes advantage of this by performing multiple literal searches simultaneously (see Section 4.3). To do this, UnMASC is split into two parts, *RuleLearner* (Section 4.2.1) and *LiteralEvaluator* (Section 4.2.2). *RuleLearner* is responsible for determining which entities to search, tracking rules and the target entity IDs satisfied by the rules. *RuleLearner* executes multiple instances of *LiteralEvaluator* which determine the best literal, given a specific set of entities to search.

Algorithm 1 RuleLearner (*minSupp*, *DB*, *TargetEntityType*, *TargetLabel*)

```
1:  RuleSet ← ∅
2:  EntityIDs ← IDs of entities of type TargetEntityType
3:  UncoveredEntityIDs ← EntityIDs
4:  while |UncoveredEntityIDs| > |EntityIDs| * minSupp          //start a new rule
5:      Rule ← empty rule
6:      RuleEntities ← EntityIDs covered by Rule
7:      while |RuleEntities| > |EntityIDs| * minSupp            //search for literal
8:          neigEntitySet ← ∅
9:          for each neigEntity with relationship to entity-type referenced in Rule
10:             neigEntitySet ← neigEntitySet + neigEntity
11:         while |neigEntitySet| > 0
12:             while all threads busy
13:                 WAIT
14:             Start Thread [ LiteralEvaluator(Rule, neigEntity, RuleEntities) ]
15:             neigEntitySet ← neigEntitySet - neigEntity
16:         BestLiteral ← result with highest FG value from all results
17:         Update Rule by adding BestLiteral
18:         RuleEntities ← remove from RuleEntities entities covered by BestLiteral
19:     UncoveredEntityIDs ← remove from UncoveredEntityIDs entities covered by Rule
20:     RuleSet ← RuleSet ∪ {Rule}
21: return RuleSet
```

Figure 10: UnMASC RuleLearner

4.2.1 RuleLearner

Initially, the rule-learning starts with an empty rule *R*, with the target-entity type τ given (Figure 10 – line 5). *RuleLearner* assembles the set *neigEntitySet* of *neigEntity* that require evaluation: if *R* is empty, then all entity types that have a neighbourhood relationship with τ are added into the set. Otherwise any entity type with a neighbourhood relationship with any referenced entity type in *R* is added to the set (lines 8-10). Once *neigEntitySet* is complete, *RuleLearner* will start to evaluate each. It does so by calling *LiteralEvaluator*, which, given the current rule and target entity IDs that satisfy the rule thus far, finds and returns the best literal (lines 11-15). A queuing system allows the number of simultaneous threads of *LiteralEvaluator* to be limited by the number of CPUs present. As a *LiteralEvaluator* completes a task, a new *neigEntity* is assigned to it.

During the literal search, once all neighbouring entity types are evaluated, the one that produces the highest FOILGain is selected and, if the resulting rule satisfies the minimum support criteria, is added to the rule (lines 16-18). If none of the refinements of a rule achieves the minimum support, then the current rule is complete. The entities in the training dataset which satisfy the rule are removed, and a new rule is started (lines 19-20).

4.2.2 LiteralEvaluator

LiteralEvaluator (Figure 11) returns the best literal and FOILGain, given a *neigEntity* and training entities not covered by any previous rule. First *LiteralEvaluator* retrieves the required dataset for analysis (lines 2-3), looking at each feature of *neigEntity*, and applying all appropriate aggregation functions (lines 4-7). For each aggregated feature, a second feature is selected and multi-feature aggregation is performed (lines 8-11). If *neigEntity* is a spatial entity, then the spatial features are extracted and spatial aggregation is performed (lines 12-17). Each aggregation result is searched for the best comparison operator and value with the highest FOILGain. The aggregation, feature and threshold with the highest FOILGain are returned (line 21).

4.3 Parallel Literal Search

RuleLearner creates multiple *LiteralEvaluator* threads simultaneously, each responsible for analyzing a different neighbourhood relationship. The number of simultaneous evaluations varies but is limited by the number of CPUs, so each runs on a dedicated CPU. If the number of possible neighbourhood relationships exceeds the available CPUs, then they are placed into a queue and are evaluated when a CPU becomes available. This setup is illustrated in Figure 12.

The different *LiteralEvaluator* threads all share the same main memory, although each thread has its own independent portion since each works with a different subset of the data. For each thread, the relevant entity-types, entities and features are retrieved from the database and loaded into main memory. Once a thread is finished with its own task, the memory is released, the result of the thread stored with the other results, and the required dataset for the next queued task is retrieved (Figure 12 - Time 2). The dataset required for each thread is computed by a join of some database tables and the time required for the retrieval is negligible since no aggregations are performed during retrieval.

There are however considerable differences between the sizes of the datasets used by different threads. For example, using the tasks from Figure 12 Time 1, the cost for evaluating {*Malls*} is expected to be relatively small since there are only a few malls in a city and no aggregations are involved (since there's only a single entity-type). {*Malls, Houses*} however is expected to incur much higher cost since there are many houses in a city, and the evaluation requires that houses be aggregated over their neighbouring malls. {*Malls, Roads*} incurs a cost that is between the other two threads since the number of roads is likely to be smaller than the number of houses in a typical city.

Since the cost for each task varies, sometimes greatly, it is possible that a very costly task is executed last, in which case all but one threads are idle and the benefits of parallelization are invalidated. The risk of this depends only on the variability of the number of entities of each entity-type: the larger the variability,

Algorithm 2 LiteralEvaluator(*rule*, *neigEntity*, *entities*)

```
1:  FG' = 0
2:  determine relationships between entities covered by rule and entities in neigEntity
3:  retrieve corresponding dataset from DB
4:  for all currFeat1 of neigEntity
5:      for each singleAgg from all single-feature aggregation functions (incl. existential)
6:          FG ← Calculate FOILGain for currFeat1 using singleAgg
7:          if FG' < FG then [currFeat1', currFeat2', Aggr', FG'] = [currFeat1, , singleAgg, FG]
8:      for all currFeat2 of neigEntity
9:          for each multiAgg from all multi-feature aggregation functions
10:             FG ← Calculate FOILGain for <currFeat1, currFeat2> using multiAgg
11:             if FG' < FG then [currFeat1', currFeat2', Aggr', FG'] = [currFeat1, currFeat2, multiAgg, FG]
12:     if neigEntity is a spatial entity
13:         extract spatial features
14:         for each spatial feature spatFeat of neigEntity
15:             for each spatAgg from all spatial aggregation functions
16:                 FG ← Calculate FOILGain for <currFeat1, spatFeat> using spatAgg
17:                 if FG' < FG then [currFeat1', currFeat2', Aggr', FG'] = [currFeat1, spatFeat, spatAgg, FG]
18:     candLit ← [neigEntity, currFeat1', currFeat2', Aggr', FG']
19: return candLit
```

Figure 11: UnMASC LiteralEvaluator (to denote best solution, an ′ is used)

the larger the differences in task-sizes. If the cost for each task can be estimated well enough, the queue can be reprioritized to avoid this scenario. Since the size of the task is unknown a priori, we estimate this based on the number of entities of each type and the number of relationships between them. Then we rearrange the tasks in the queue such that the largest tasks get executed first.

Each search is made up of a permutation of n entity types which can be denoted as $\{t_1, \ldots, t_{k-1}, t_k, \ldots, t_n\}$, where $t_1 = \tau$. Each entity type t_{k-1} has, on average, $| t_{k-1} \rhd\lhd t_k | / | t_{k-1} |$ neighbours of type t_k, where $| t_{k-1} \rhd\lhd t_k |$ denotes the number of relationships between all entities of type t_{k-1} and t_k, and $| t_{k-1} |$ denotes the number of entities of type t_{k-1}. For example, assume there are 10 malls ($| t_{k-1} |$) and in total 100 neighbourhood relationships between malls and houses ($| t_{k-1} \rhd\lhd t_k |$), then this implies that, on average, there are 100/10 = 10 neighbouring houses per mall. Thus the total number of relationships between the target entity-type (t_1) and the entity-type being searched (t_n) can be estimated by

$$\text{cost}(t_1, t_n) = \prod_{k=2}^{n} \frac{|t_{k-1} \rhd\lhd t_k|}{|t_{k-1}|}.$$

Multi-feature and spatial aggregation is then performed on the features (f) of the entity-type that is searched (t_n) by choosing two features to aggregate, which can be done in ${}_f C_2 = {}^{f!}\!/_{2!(f-2)!}$ ways. Thus the total cost of a specific task is estimated by

$$\text{total\_cost}(t_1, t_n) = |t_k| \frac{f!}{2(f-2)!} \prod_{k=2}^{n} \frac{|t_{k-1} \rhd\lhd t_k|}{|t_{k-1}|}$$

5. EXPERIMENTS

As a result of collaboration with the Criminology Department at Simon Fraser University (SFU), real-world crime data was available with location, time, and type of calls for service for the Royal Canadian Mounted Police (RCMP) in British Columbia (BC) between August 1, 2001 and August 1, 2006. Also available was the British Columbia Assessment Authority (BCAA) dataset containing the property values of all plots of land within BC. The city of Burnaby, BC, was selected as the test dataset. As a pre-processing step, each plot was assigned a counter, denoting the number of burglaries which have occurred at that location. This count was then used to create a Boolean class-label (if *count>0* then *burglarized*, else *not burglarized*). The entire dataset contains 22 different entity types, with each type containing from 3 to 34,000 entities, an average of 3,100 entities per type.

UnMASC was evaluated using three neighbourhood definitions. With the first alternative, the neighbours were pre-materialized using buffer zones of variable size, where the radius is given by:

$$radius = \sqrt{area(Burnaby) / \{\pi \times count(EntityType)\}}$$

This dataset contained 2.8 million spatial relationships between entities. The second dataset was pre-materialized using the Voronoi neighbourhoods proposed in this paper, with 3.8 million spatial relationships. In order to evaluate the importance of the neighbourhood, a third was created which contained only the target entity-type. Note that the third alternative only uses information on the target entity type and since no neighbouring entity types are evaluated, the search-space is much smaller. All three methods were tested and compared for their impact on precision, recall and accuracy. Measurements were taken for the rules learnt on the target class (*burglarized*).

Experiments were run using an implementation of the UnMASC algorithm running on an 8-CPU Windows server tied to a backend database with DB/2 v9.1 Spatial Extender. The operating system was allowed to set the CPU-affinity of each instance, which meant no two instances were running on the same CPU simultaneously. 5-fold cross-validation was performed. To evaluate the effectiveness of the parallelization of UnMASC, classification was performed in both parallel (6 threads) and serial (1 thread). Note that for the dataset with only the target entity-type, since there is only a single type, producing only a single task for each literal search, this dataset could not be evaluated in a parallel fashion.

5.1 Burglaries of Commercial Properties

For our first experiment, the set of 2812 commercial properties were selected as the target entities. Commercial properties where

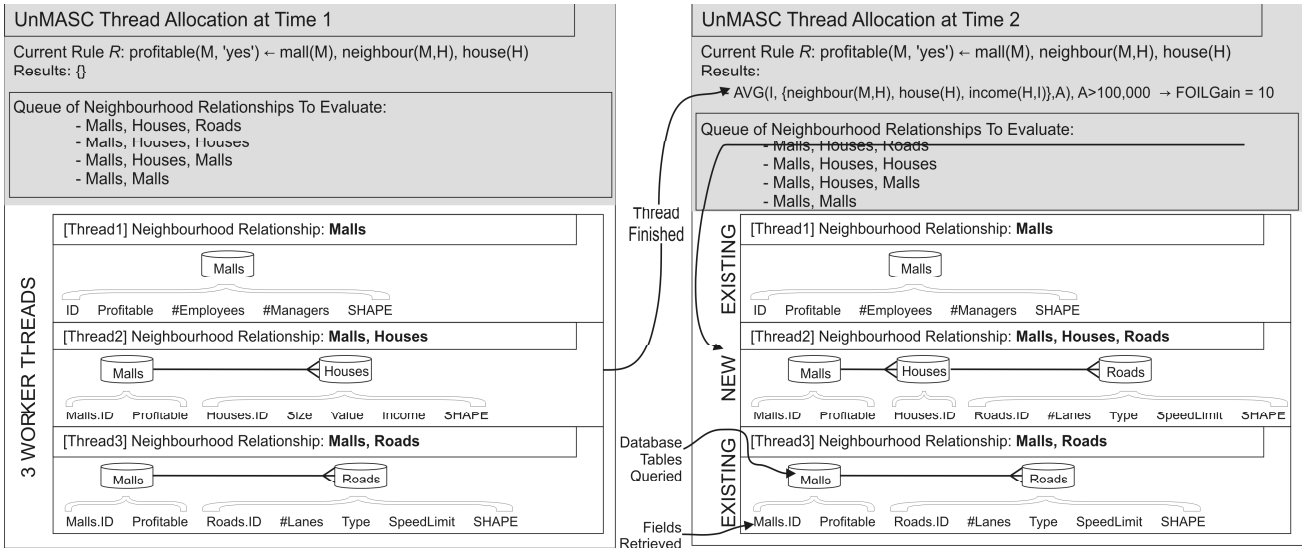


Figure 12: Search Process Example

burglaries have occurred were chosen as the target-label, with 33% of the properties falling under this label within the 5-year period considered. The Voronoi neighbourhood definition had higher precision while having lower recall. Accuracy was higher. The parallel version of UnMASC outperformed the serial version by a factor of about 5.4 (Figure 13). Rules built only on the target entity-type, without any neighbouring entities, performed barely better than the trivial classifier. This clearly illustrates the significance the neighbourhood plays in spatial classification.

It is interesting to note that there was a significant difference between buffer zones and Voronoi neighbourhoods in the number of neighbours each commercial property has. On average, for each commercial property, the number of neighbouring commercial properties using buffer zones was 20 times that of Voronoi neighbourhoods, while the number of neighbouring non-commercial properties was only 0.4 times. This clearly indicates that buffer zone rules were built more on neighbouring commercial properties and less on non-commercial properties, than the rules built on Voronoi neighbourhoods. Due to the use of zoning for city planning in Canada, commercial properties (as well as other property types) tend to be clustered, which the buffer zone is unable to bypass and hence precision of the resulting rules suffers. This was also the reason for the large difference in run-times: when performing the aggregations, the buffer zone had to aggregate much fewer values than the Voronoi approach, resulting in a reduction in runtime (and precision). Note that this difference in runtime was not seen in the next set of experiments.

5.2 Burglaries of Industrial Properties

As a second classification task, 453 industrial properties were selected for rule learning with the 16% of properties which were burglarized selected as the target. Precision and recall for Voronoi neighbourhoods were 57% and 69.4% respectively, beating the buffer zone by 4% and the trivial classifier by 3%. Both neighbourhood approaches had comparable runtimes, in both parallel and serial mode, with the parallel version of UnMASC outperforming the serial version by, on average, a factor of 5.1 (Figure 13). Rules built only on the target entity-type performed somewhat better than the trivial classifier but significantly worse than rules built with neighbouring entity-types.

5.3 Burglaries of High Rise Properties

As a third classification task, 1036 high-rise properties were selected for learning rules with the 44.3% of properties which were burglarized selected as the target. Precision and recall for Voronoi neighbourhoods were 85.2% and 87.8% respectively, beating the buffer zone by a small margin, and the trivial classifier's accuracy (55.7%) by 32%. Both neighbourhood approaches had comparable runtimes, with the parallel version of UnMASC outperforming the serial version by, on average, a factor of 5.21 (Figure 13). Similar to the previous experiments, including neighbouring entity-types provided a significant increase in precision and accuracy, demonstrating again the importance of the role of neighbourhood.

5.4 Rule Analysis

R_7 (Figure 14) illustrates a sample rule that was found. It denotes that a commercial property (L1) is likely to be burglarized if the median building-value of neighbouring industrial properties is worth less than $445,000 (L2) and has neighbouring parks with areas less than 14,400m^2 (L4). The commercial property also has neighbouring industrial properties with a spatial trend where duplexes are cheaper the further east they are, and more expensive the further west they are. This could indicate that the commercial property is in a relatively inexpensive industrial neighbourhood, acting as an attractor to crime, while neighbour to parks which could be a source of people inclined to commit crimes.

Another interesting sample rule involving high-rises, R_8 (Figure 14), was discovered. It states that a high-rise is likely to be burglarized if it is worth more than $2.2 million (L1), the average distance to a commercial property is less than 302 m (L2) and, for neighbouring commercial properties (L3), neighbouring parks tend to be small (L4), and duplexes tend to be worth at least $351,500. This rule seems to indicate that expensive high-rises near high-traffic areas are going to see more burglaries as opposed to ones in areas without traffic.

6. CONCLUSION

In this paper a multi-relational approach to spatial classification was presented and novel challenges identified. A Voronoi-diagram based neighbourhood definition was proposed to make explicit the spatial relationships. In order to handle spatial features and aggregations in First Order Logic, a formal set of additions to the framework was introduced. A scalable implementation was presented, capable of evaluating multiple literal candidates simultaneously, thus significantly decreasing the runtime required for rule-learning. Experiments on a real-world spatial dataset showed substantial gains in precision, recall and accuracy compared to existing approaches for neighbourhood definitions. Experimentation without including neighbourhood

Target Type	Class Distribution		Neighbourhood	Precision	Recall	Accuracy	Runtime (Parallel)	Runtime (Serial)	Speed-up
Commercial	931+	1881-	None	41.6%	**93.7%**	54.3%		0h:37m	
			Buffer Zone	56.4%	92.5%	73.8%	3h:40m	19h:32m	5.32x
			Voronoi	**65.9%**	86.7%	**80.6%**	7h:56m	43h:05m	5.44x
Industrial	73+	380-	None	19.8%	**77.1%**	45.2%		0h:13m	
			Buffer Zone	53.1%	57.6%	84.1%	4h:05m	20h:25m	5.00x
			Voronoi	**57.0%**	69.4%	**86.3%**	3h:47m	19h:32m	5.16x
High Rises	459+	577-	None	57.1%	**97.6%**	66.4%		0h:4m	
			Buffer Zone	83.4%	87.7%	86.3%	3h:18m	17h:06m	5.18x
			Voronoi	**85.2%**	87.8%	**87.7%**	4h:31m	23h:37m	5.23x

Figure 13: Burglaries of Commercial, Industrial and High-Rise Housing Properties

R_7: burglarized(C, 'yes') ←	commercial(C),	L1
	MEDIAN(B_1, {industrial(I), neighbour(C,I), building_value(I,B_1)}, M), M<445,000	L2
	industrial(I), neighbour(C,I), TREND({B_2, X}, {duplex(D), neighbour(I,D), building_value(D, B_2), x_coord(D,X)}, S), S>0.798	L3
	MAX(A, {park(P), neighbour(C,P), area(P,A)}, N), N<14,400	L4
R_8: burglarized(H, 'yes') ←	highrise(H), building_value(H,V), V>2,160,000	L1
	AVG(D1, {commercial(C), neighbour(H,C), distance(H,C,D1)}, A), A<302	L2
	commercial(C), neighbour(H,C)	L3
	MEDIAN(W, {park(P), neighbour(C,P), width(P)}, M_1), M_1<122	L4
	MEDIAN(X, {duplex(D), neighbour(C,D), land_value(D)}, M_2), M_2>351,500	L5

Figure 14: Sample rules discovered by UnMASC

information resulted in low accuracy and precision, demonstrating the importance of selecting the proper neighbourhood definition. Anecdotal evidence was provided to illustrate the meaningfulness of the rules discovered and the expressiveness of the language.

For future work, we plan to optimize the parallelization by looking for ways to break each job into smaller pieces, perhaps where the literal is not searched for in an entity-type, but only a feature of an entity-type. Secondly, it would be interesting to incorporate the temporal aspects into rule-learning to detect, for example, temporal trends and how changes over time in neighbourhood composition can lead to better predictions.

7. REFERENCES

[1] Adam, N. R; Janeja, V. P.; Atluri, V.: "Neighborhood based detection of anomalies in high dimensional spatio-temporal sensor datasets", In Proceedings of the ACM symposium on Applied computing (2004)

[2] Appice, A.; Ceci, M.; Lanza, A.: "Discovery of spatial association rules in geo-referenced census data: a relational mining approach", In Proceedings of Intelligent Data Analysis (2003)

[3] Bembenik, R.; Rybinski, H.: "Mining Spatial Association Rules with no Distance Parameter", Advances in Soft Computing, Vol. 35 (2006)

[4] Ceci, M.; Appice, A.; Malerba, D.: "Spatial Associative Classification at Different Levels of Granularity: A Probabilistic Approach", Lecture Notes in Computer Science, Volume 3202 (2004)

[5] Ceri, S; Gottlob, G; Tanca, L.: "Logic programming and databases", Springer-Verlag, (1990)

[6] Chelghoum, N., Zeitouni K.: "Spatial Data Mining Implementation - Alternatives And Performances", GeoInfo, pp 127-153 (2004)

[7] Egenhofer, M.J.: "Reasoning about Binary Topological Relations", In Proceedings of SSD, (1991)

[8] Estivill-Castro, V.; Lee, I.: "Argument free clustering for large spatial point-data sets via boundary extraction from Delaunay Diagram", Computers, Environment and Urban Systems, Vol. 26, No 4, pp. 315-334 (July 2002)

[9] Frank, R., Moser, F., Ester, M.: "A Method for Multi-Relational Classification Using Single and Multi-Feature Aggregation Functions", In Proceedings of PKDD (2007)

[10] Hoff, K.; Culver, T.; Keyser, J.; Lin, M.; Manocha, D.:"Fast computation of generalized Voronoi diagrams using graphics hardware", In Proceedings of SIGGRAPH (1999)

[11] Knobbe, A.J.; Siebes, A.; Marseille, B.: "Involving Aggregate Functions in Multi-Relational Search", In Proceedings of PKDD (2002)

[12] Longley, P.A.; Goodchild, M.F.; Maguire, D.J; Rhind, D.W.: "Geographical Information Systems: Principles, Techniques, Applications and Management", Wiley, 2nd Edition (1999)

[13] Miller, H. J.: "Tobler's First Law and spatial analysis", Annals of the Association of American Geographers, 94. (2004)

[14] Ohyama ,T.: "Some Voronoi diagrams that consider consumer behavior analysis", In Industrial Mathematics of the Japan Journal of Industrial and Applied Mathematics (2005)

[15] Papadias, D., Theodoridis, Y.: "Spatial Relations, Minimum Bounding Rectangles, and Spatial Data Structures", International Journal of Geographic Information Science Volume 11, Issue 2 (1997)

[16] Quinlan, J. R., Cameron-Jones, R. M.: FOIL – A midterm report, In Proceedings of ECML (1993)

[17] Santos, M. Y.; Amaral L. A.: "Geo-spatial data mining in the analysis of a demographic database", In Soft Computing - A Fusion of Foundations, Methodologies and Applications, Volume 9, Issue 5 (2005)

[18] Schlossberg, M.: "GIS, The US Census And Neighbourhood Scale Analysis", Planning, Practice & Research, Vol. 18, No. 2–3 (2003)

[19] Taskar, B., Segal, E., and Koller, D.: "Probabilistic classification and clustering in relational data", In Proceedings IJCAI (2001)

[20] Tobler, W.R.: "A computer movie simulating urban growth in the Detroit region", In Economic Geography, Volume 46 (1970)

[21] Vens, C.; Van Assche, A.; Blockeel, H.; Dzeroski, S.: "First order random forests with complex aggregates", In Proceedings of ILP (2004)

[22] Yin, X.; Han J.; Yang J.; Yu, P.S.: "CrossMine: Efficient Classification Across Multiple Database Relations", In Proceedings of ICDE (2004)