Goal: Provide an introduction to some spatial mining techniques.

- Introduction
- Spatial Data Overview
- Spatial Data Mining Primitives
- Generalization/Specialization
- Spatial Rules
- Spatial Classification
- Spatial Clustering
Spatial Object

- Contains both spatial and nonspatial attributes.
- Must have a location type attributes:
  - Latitude/longitude
  - Zip code
  - Street address
- May retrieve object using either (or both) spatial or nonspatial attributes.
Spatial Data Mining Applications

- Geology
- GIS Systems
- Environmental Science
- Agriculture
- Medicine
- Robotics
- May involve both spatial and temporal aspects
Spatial Queries

- Spatial selection may involve specialized selection comparison operations:
  - Near
  - North, South, East, West
  - Contained in
  - Overlap/intersect

- **Region (Range) Query** – find objects that intersect a given region.

- **Nearest Neighbor Query** – find object close to identified object.

- **Distance Scan** – find object within a certain distance of an identified object where distance is made increasingly larger.
Spatial Data Structures

- Data structures designed specifically to store or index spatial data.
- Often based on B-tree or Binary Search Tree
- Cluster data on disk based on geographic location.
- May represent complex spatial structure by placing the spatial object in a containing structure of a specific geographic shape.

Techniques:
- Quad Tree
- R-Tree
- k-D Tree
Minimum Bounding Rectangle
Smallest rectangle that completely contains the object
MBR Examples

a) Lake

<x1,y1>

b) MBR for Lake

<x2,y2>

c) Smaller MBRs for Lake
Quad Tree

- Hierarchical decomposition of the space into quadrants (MBRs)
- Each level in the tree represents the object as the set of quadrants which contain any portion of the object.
- Each level is a more exact representation of the object.
- The number of levels is determined by the degree of accuracy desired.
Quad Tree Example

a) Representing Triangle With Quadrants

b) Quad Tree
R-Tree

- As with Quad Tree the region is divided into successively smaller rectangles (MBRs).
- Rectangles need not be of the same size or number at each level.
- Rectangles may actually overlap.
- Lowest level cell has only one object.
- Tree maintenance algorithms similar to those for B-trees.
R-Tree Example

a) Partitioning with MBRs

b) R-Tree
K-D Tree

- Designed for multi-attribute data, not necessarily spatial
- Variation of binary search tree
- Each level is used to index one of the dimensions of the spatial object.
- Lowest level cell has only one object
- Divisions not based on MBRs but successive divisions of the dimension range.
k-D Tree Example

a) Divide and Conquer Partitioning

b) k-D Tree
Topological Relationships

- Disjoint
- Overlaps or Intersects
- Equals
- Covered by or inside or contained in
- Covers or contains
Distance Between Objects

- Euclidean
- Manhattan
- Extensions:

- **minimum.**
  \[
  dis(A, B) = \min_{(x_a, y_a) \in A, (x_b, y_b) \in B} dis((x_a, y_a), (x_b, y_b))
  \]

- **maximum.**
  \[
  dis(A, B) = \max_{(x_a, y_a) \in A, (x_b, y_b) \in B} dis((x_a, y_a), (x_b, y_b))
  \]

- **average.**
  \[
  dis(A, B) = \text{average}_{(x_a, y_a) \in A, (x_b, y_b) \in B} dis((x_a, y_a), (x_b, y_b))
  \]

- **center.**
  \[
  dis(A, B) = dis((x_{ca}, y_{ca}), (x_{cb}, y_{cb}))
  \]

  where \((x_{ca}, y_{ca})\) is a center point for object \(A\).
Progressive Refinement

- Make approximate answers prior to more accurate ones.
- Filter out data not part of answer
- Hierarchical view of data based on spatial relationships
- Coarse predicate recursively refined
Progressive Refinement

Diagram:

- Dallas-Fort Worth Metroplex
  - Forth Worth
  - Dallas
  - Arlington
  - Mid-Cities
  - Northern Suburbs
  - Park Cities
    - Preston Hollow
    - M Streets
    - Lakewood
    - East
    - University Park
    - Highland Park
Spatial Data Dominant Algorithm

Input:

- $D$ //Spatial database.
- $H$ //Spatial hierarchy.
- $C$ //Concept hierarchy.
- $q$ //Query.

Output:

- $R$ // Rule which states the general characteristics requested.

SPATIAL-DATA-DOMINANT Algorithm

- $D' = \text{set of data obtained from } D \text{ based on selection criteria in } q$;
- Following the structure of $H$, combine data into regions until either the desired threshold number of regions is found or the requested level in $H$ is obtained;
- for each region found do
  - perform an attribute-oriented induction on the nonspatial attributes;
  - Generate and out a rule which summarizes the results found;
STING

- STatistical Information Grid-based
- Hierarchical technique to divide area into rectangular cells
- Grid data structure contains summary information about each cell
- Hierarchical clustering
- Similar to quad tree
STING

a) Level 1

b) Level 2

c) Level 3
STING Build Algorithm

Input:
\[ D \] // Data to be placed in the hierarchical structure.
\[ k \] // Number of desired cells at the lowest level.

Output:
\[ T \] // Tree.

STING BUILD Algorithm

// Create empty tree from top down.
\[ T = \text{root node with data values initialized}; \] // Initially only root node
\[ i = 1; \]
repeat
    for each node in level \( i \) do
        create 4 children nodes with initial values;
        \[ i = i + 1; \]
    until \( 4^i = k; \)
// Populate tree from bottom up.
for each item in \( D \) do
    determine leaf node \( j \) associated with the position of \( D; \)
    update values of \( j \) based on attribute values in item;
    \[ i := \log_4(k); \]
repeat
    \[ i := i - 1; \]
    for each node \( j \) in level \( i \) do
        update values of \( j \) based on attribute values in its 4 children;
    until \( i = 1; \)
STING Algorithm

Input:
\( T \) // Tree.
\( q \) // Query.

Output:
\( R \) // Regions of relevant cells.

STING Algorithm

\( i = 1; \)

repeat
    for each node in level \( i \) do
        determine if this cell is relevant to \( q \) and mark as such;
        \( i = i + 1; \)
    until all layers in the tree have been visited;

identify neighboring cells of relevant cells to create regions of cells;
Spatial Rules

- **Characteristic Rule**
  The average family income in Dallas is $50,000.

- **Discriminant Rule**
  The average family income in Dallas is $50,000, while in Plano the average income is $75,000.

- **Association Rule**
  The average family income in Dallas for families living near White Rock Lake is $100,000.
Spatial Association Rules

- Either antecedent or consequent must contain spatial predicates.
- View underlying database as set of spatial objects.
- May create using a type of progressive refinement
## Spatial Association Rule Algorithm

**Input:**
- $D$ // Data, including spatial and nonspatial attributes
- $C$ // Concept hierarchies
- $s$ // Minimum support for levels.
- $\alpha$ // Confidence
- $q$ // Query to retrieve interested objects
- $P$ // Topological predicate(s) of interest

**Output:**
- $R$ // Spatial Association Rules.

**SPATIAL ASSOCIATION RULE Algorithm**

1. $D' = q(D)$;
2. $CP$ is built by applying the coarse predicate version of $P$ to $D'$;
3. $CP$ consists of the set of coarse predicates satisfied by pairs of objects in $D'$.
4. Determine the set of frequent coarse predicates $FCP$
5. By finding the coarse predicates which satisfy $s$;
6. Find the set of frequent fine predicates $FFP$ from $FCP$;
7. Find $R$ by finding all frequent fine predicates and then generating rules;
Spatial Classification

- Partition spatial objects
- May use nonspatial attributes and/or spatial attributes
- Generalization and progressive refinement may be used.
ID3 Extension

- Neighborhood Graph
  - Nodes – objects
  - Edges – connects neighbors
- Definition of neighborhood varies
- ID3 considers nonspatial attributes of all objects in a neighborhood (not just one) for classification.
Spatial Decision Tree

- Approach similar to that used for spatial association rules.
- Spatial objects can be described based on objects close to them – *Buffer*.
- Description of class based on aggregation of nearby objects.
Spatial Decision Tree Algorithm

Input:
\[ D \] // Data, including spatial and nonspatial attributes
\[ C \] // Concept hierarchies

Output:
\[ T \] // Binary Decision Tree.

SPATIAL DECISION TREE Algorithm

find a sample \( S \) of data from \( D \) with known classification;
identify the best predicates \( p \) to use for classification;
determine the best buffer size and shape;
using \( p \) and \( C \), generalize the predicates for each buffer;
build binary \( T \) using the generalized predicates and ID3;
Spatial Clustering

- Detect clusters of irregular shapes
- Use of centroids and simple distance approaches may not work well.
- Clusters should be independent of order of input.
Spatial Clustering
CLARANS Extensions

- Remove main memory assumption of CLARANS.
- Use spatial index techniques.
- Use sampling and R*-tree to identify central objects.
- Change cost calculations by reducing the number of objects examined.

*Voronoi Diagram*
Voronoi

a) Perpendicular Bisector

b) Voronoi Polyhedrons
SD(CLARANS)

- Spatial Dominant
- First clusters spatial components using CLARANS
- Then iteratively replaces medoids, but limits number of pairs to be searched.
- Uses generalization
- Uses a learning to derive description of cluster.
SD(CLARANS) Algorithm

Input:
- $D$ // Data to be clustered.
- $k$ // Number of desired cells at the lowest level.

Output:
- $K$ // Set of Clusters.

SD(CLARANS) Algorithm

// Find set of tuples which satisfy selection criteria.
$D' = \text{select tuples from } D \text{ based on nonspatial selection criteria;}

// Apply CLARANS to $D'$ based on spatial attributes.
$K = \text{CLARANS}(D')$;

// Perform attribute generalization.
for each $k \in K$ do
  apply DBLEARN to the nonspatial attributes in $k$;
DBCLASD

- Extension of DBSCAN
- Distribution Based Clustering of LArge Spatial Databases
- Assumes items in cluster are uniformly distributed.
- Identifies distribution satisfied by distances between nearest neighbors.
- Objects added if distribution is uniform.
DBCLASD Algorithm

Input:

\( D \)  // Spatial objects to be clustered.

Output:

\( K \)  // Set of Clusters.

DBCLASD Algorithm

\( k = 0; \)  // Initially there are no clusters.

\( c = \emptyset; \)  // Initialize the set of candidates to be empty.

for each point \( p \) in \( D \) do

if \( p \) is not in a cluster then

create a new cluster \( C \) and put \( p \) in \( C \);

add neighboring points of \( p \) to \( C \);

for each point \( q \) in \( C \) do

add the points in the neighborhood of \( q \) which have not been processed to \( c \);

expand \( C \);
Aggregate Proximity

- **Aggregate Proximity** – measure of how close a cluster is to a feature.
- Aggregate proximity relationship finds the k closest features to a cluster.
- **CRH Algorithm** – uses different shapes:
  - Encompassing Circle
  - Isothetic Rectangle
  - Convex Hull