

Context-Based Classification of Objects in Topographic Data

Leo Mulhare B.Sc.

Submitted in fulfilment of the degree of M.Sc.



NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

National University of Ireland, Maynooth

Department of Computer Science

October 2007

Head of Department: Dr. Adam C. Winstanley

Research Supervisor: Dr. Adam C. Winstanley

Table of Contents

1: INTRODUCTION	1
1.1 LARGE-SCALE TOPOGRAPHIC DATABASES.....	1
1.2 CREATING & CLASSIFYING AREA FEATURES	4
1.3 CONTEXT-BASED CLASSIFICATION USING STRUCTURE TEMPLATE MATCHING	7
1.4 AIMS OF RESEARCH.....	8
1.5 STRUCTURE OF THIS DISSERTATION	8
2: LARGE-SCALE TOPOGRAPHIC DATABASES	9
2.1 GEOGRAPHIC INFORMATION SYSTEMS	9
2.1.1 <i>The Geographical Co-ordinate System</i>	9
2.1.2 <i>Map Projections and the Cartesian Co-ordinate System</i>	10
2.1.3 <i>Raster Data Model</i>	10
2.1.4 <i>Cartographic Symbolisation</i>	12
2.1.5 <i>Cartographic Generalisation</i>	12
2.2 VECTOR DATA MODEL	13
2.2.1 <i>Vector Data Layers</i>	14
2.3 ORDNANCE SURVEY IRELAND	15
2.4 ORDNANCE SURVEY (GREAT BRITAIN)	15
2.4.1 <i>OS MasterMap</i>	15
2.4.1.1 TOIDS	16
2.4.1.2 A Polygon Layer.....	16
2.4.1.3 Topological Polygons	16
2.4.1.4 Seamless Coverage	16
2.4.1.5 Change-Only Updates.....	16
2.4.1.6 GML-encoded.....	17
2.4.1.7 SVG Compatibility	17
2.5 CHOICE OF OS MASTERMAP AS TEST CASE DATA.....	18

3: ANALOGY	19
3.1 DEFINITION OF ANALOGY.....	19
3.2 METAPHOR & ANALOGY.....	20
3.3 ANALOGY IN LEARNING.....	23
3.4 LETTER STRING ANALOGIES.....	23
3.5 GEOMETRIC ANALOGIES.....	24
3.6 PREDICATE REPRESENTATION OF ANALOGICAL DOMAINS.....	26
3.7 ANALOGICAL DOMAIN STRUCTURE & GRAPH REPRESENTATION.....	27
3.8 STRUCTURE MAPPING.....	29
3.9 PHASES OF ANALOGY.....	30
3.9.1 Representation.....	30
3.9.2 Retrieval.....	30
3.9.3 Mapping.....	31
3.9.4 Validation.....	31
3.9.5 Induction.....	32
3.10 SUMMARY.....	33
4: TOPOGRAPHICAL ANALOGIES	34
4.1 INTRODUCTION.....	34
4.2 COMPARISON OF GEOMETRIC ANALOGIES AND TOPOGRAPHIC DATA.....	35
4.2.1 Objects.....	35
4.2.2 Attributes.....	36
4.2.3 Relations.....	37
4.3 APPLYING GEOMETRIC ANALOGY PROBLEMS TO TOPOGRAPHIC DATA.....	38
4.4 CONTEXTUAL ANALYSIS OF AREA FEATURES.....	39
4.4.1 Types of Adjacency.....	41
4.4.2 Graph Representation of Context.....	42
4.5 AN ANALOGICAL CLASSIFICATION MODEL FOR TOPOGRAPHIC DATA.....	43
4.5.1 Context Isomorphism.....	43
4.5.2 Context-Based Classification.....	44
4.5.3 Comparison of Geometric and Topographic Analogies.....	45
4.5.4 Templates and Ambiguity.....	46
4.5.5 Frequency Templates.....	47
4.6 CONTENT VECTOR-BASED CLASSIFICATION.....	48
4.7 COMBINING CONTENT VECTOR AND CONTEXT STRUCTURE CLASSIFICATION.....	50
4.8 CONCLUSION.....	52

5: IMPLEMENTATION	53
5.1 INTRODUCTION	53
5.2 GIS EMPLOYED	53
5.2.1 Topological Connectedness	53
5.2.2 Topological Data	54
5.2.3 Non-Topological Data	54
5.2.4 ESRI ArcView GIS and the Shapefile Format	55
5.2.5 Optimizing the Performance of Spatial Indexing in ArcView GIS	56
5.3 PREPROCESSING	57
5.3.1 GML to Shapefile Conversion	57
5.3.2 Cartographic Polygon Removal	57
5.3.3 Duplicate Feature Removal	58
5.3.4 Perimeter Polygon Removal	59
5.4 CONTEXT EXTRACTION	61
5.4.1 Data Structure Representation of Polygonal Context	61
5.4.1.1 The Adjacency Matrix	61
5.4.1.2 Prolog Implementation	63
5.4.1.3 Atomic Predicate Representation of Context	64
5.4.1.4 Complex Predicate Representation of Context	66
5.5 CLASSIFICATION TEMPLATE CONSTRUCTION	67
6: RESULTS	69
6.1 INTRODUCTION	69
6.2 TRAINING AND TESTING DATA SETS USED	69
6.2 PROBABILISTIC CLASSIFICATION VIA CVM AND CSM	73
6.3 COMBINING CVM AND CSM CLASSIFICATION RESULTS	74
6.4 TEMPLATE QUALITY (TRAINING RESULTS)	75
6.5 CLASSIFICATION RESULTS (TESTING RESULTS)	78
6.5.1 Analysis of Classification Results	79
6.6 VISUALIZATIONS OF CLASSIFICATIONS	84
6.6 CONCLUSION	88

7: CONCLUSION	89
7.1 INTRODUCTION	89
7.2 EVALUATION OF CONTEXT-BASED CLASSIFICATION	89
7.2.1 <i>Dependence on Pre-Classified Data</i>	89
7.2.2 <i>Classification Template Quality</i>	90
7.2.3 <i>Unmatched Features</i>	91
7.2.4 <i>Classification Accuracy</i>	91
7.2.5 <i>Combined Classifier Performance</i>	92
7.2.6 <i>Summary & Recommendations</i>	92
7.3 IMPORTANCE OF LAND USE	93
7.4 FUTURE WORK	94
7.4.1 <i>Combining Output with Other Classifiers</i>	94
7.4.2 <i>Improving Accuracy</i>	94
7.4.3 <i>Searching For Known Classification Artifacts</i>	95
7.4.4 <i>Extending Context Model beyond Immediate Neighbours</i>	95
7.4.5 <i>Incorporation of Other Topographic Layers in Context Model</i>	96
7.5 APPLICATIONS OF WORK	97
7.5.1 <i>Topographic Vector Line Data</i>	97
7.5.2 <i>Small-Scale Topographic Data</i>	97
7.5.3 <i>Raster Data Classification</i>	98
7.5.4 <i>Classification of Features in Architectural Plans</i>	98
7.5.5 <i>Verifying Circuit Diagrams</i>	99
7.6 CONCLUSION	99
BIBLIOGRAPHY	100

Acknowledgements

We would like to thank Ordnance Survey (Great Britain) for their help and the provision of the data used throughout this project.

I am especially grateful to Adam for his assistance and support.

We are obliged to Diarmuid O'Donoghue for his collaboration in this work.

I am grateful to many in the Computer Science Department over the years for their assistance and friendship.

I would like to thank my family for their love and support.

Contributing Publications

- Mulhare, L., O'Donoghue, D. and Winstanley, A. C.** Analogical Structure Matching on Cartographic Data, *12th Irish Conference on Artificial Intelligence and Cognitive Science*, Maynooth, 43-54, 2001.
- Mulhare, L., O'Donoghue, D. and Winstanley, A. C.** Context-based Classification of Objects in Cartographic Data. 195-198, *GIS Research UK*, Sheffield, April 2002.
- O'Donoghue, D., Winstanley, A., Mulhare, L. and Keyes, L.** Applications of Cartographic Structure Matching, *IEEE International Geoscience and Remote Sensing Symposium (IGARSS'03)*, Vol. VI: 3730 – 3732, July 2003.
- O'Donoghue, D., Winstanley, A. and Mulhare, L.** Structure Matching in Cartographic Data using Analogical Reasoning, NUIM Computer Science technical report, NUIM-CS-TR-2001-07, June 2001.

Abstract

Large-scale topographic databases model real world features as vector data objects. These can be point, line or area features. Each of these map objects is assigned to a descriptive class; for example, an area feature might be classed as a building, a garden or a road. Topographic data is subject to continual updates from cartographic surveys and ongoing quality improvement. One of the most important aspects of this is assignment and verification of class descriptions to each area feature. These attributes can be added manually, but, due to the vast volume of data involved, automated techniques are desirable to classify these polygons.

Analogy is a key thought process that underpins learning and has been the subject of much research in the field of artificial intelligence (AI). An analogy identifies structural similarity between a well-known source domain and a less familiar target domain. In many cases, information present in the source can then be mapped to the target, yielding a better understanding of the latter. The solution of geometric analogy problems has been a fruitful area of AI research. We observe that there is a correlation between objects in geometric analogy problem domains and map features in topographic data. We describe two topographic area feature classification tools that use descriptions of neighbouring features to identify analogies between polygons: content vector matching (CVM) and context structure matching (CSM). CVM and CSM classify an area feature by matching its neighbourhood context against those of analogous polygons whose class is known.

Both classifiers were implemented and then tested on high quality topographic polygon data supplied by Ordnance Survey (Great Britain). Area features were found to exhibit a high degree of variation in their neighbourhoods. CVM correctly classified 85.38% of the 79.03% of features it attempted to classify. The accuracy for CSM was 85.96% of the 62.96% of features it tried to identify. Thus, CVM can classify 25.53% more features than CSM, but is slightly less accurate. Both techniques excelled at identifying the feature classes that predominate in suburban data. Our structure-based classification approach may also benefit other types of spatial data, such as topographic line data, small-scale topographic data, raster data, architectural plans and circuit diagrams.

1: Introduction

1.1 Large-Scale Topographic Databases

Since ancient times, man has made 2-dimensional pictorial representations of the landscape around him. These maps were used to help navigate, manage, settle and conquer the land which has been among the most valued assets of countless civilizations and cultures. The earliest known map was found near Kirkuk in Iraq and dates from circa 2500-2300 B.C. (Harley & Woodward, 1987). As early cartographers did not have means to accurately measure latitude, longitude or elevation, scale and direction within individual early charts tended to be inconsistent. Copies of maps were made by hand, with each subsequent generation from the original being subject to increasing variation in the aforementioned values, along with errors of omission, addition and misspellings. In time, the invention of the printing press by Johann Gutenberg in 1452 would allow for the cheap reproduction of identical maps, solving the problem of replicative fading.

These early maps, originating on paper, were all *raster-based*, meaning that they were images recorded at a fixed resolution or scale, analogous to a bitmap image in a computer. The most detailed raster maps are referred to as *large-scale*, denoting that a unit of distance on the map is equivalent to a relatively small number of units “on the ground”, allowing greater detail to be shown. A small-scale colour raster map is shown in Figure 1.1.



Figure 1.1: Section of the Ordnance Survey (Great Britain) 1:50,000 Scale Colour Raster map, showing Yarmouth on the Isle of Wight.

An alternative to the raster-based data model is the *vector-based* data model, whereby the Cartesian co-ordinates of features in the landscape, and associated attributes are stored in a database. Normally only the x and y co-ordinates (*easting* and *northing*) are recorded. The notion of scale is meaningless in relation to vector-based data, as it may be displayed at any resolution. Nevertheless, the term *Large-Scale Topographic Databases* is used to denote vector-based maps which are surveyed with greater accuracy and which include features that are omitted at smaller scales. For instance, urban landscapes are generally surveyed in the greatest detail and may have a scale of 1:1,250.

A large-scale topographic database organizes features in several *layers*. A typical set of layers might include:

points

These represent small objects whose orientation is often unimportant e.g. post-box, electricity pole. A single Cartesian co-ordinate is recorded.

addresses / text labels

These are points with associated text values and may be the name of an area, a street, or a housing estate, a house number etc. The point is only used to anchor the text spatially.

lines

These represent boundaries between areas and the centre-line of linear features that are not wide enough to have their edges represented. Some examples are the outline of a building, the boundary wall between a pair of semi-detached houses, a fence, and the edge between a road and a foot-path. A line is a sequence of *vertices* which are assumed to be connected by straight *edges*. A vertex, like a point, is a Cartesian co-ordinate, but whereas a point has attributes associated with it, a vertex is only a constituent of a larger feature.

administrative boundaries

These lines represent divisions between political areas. Some may coincide with features in the line layer such as road and river edges and field boundaries; others do not follow any apparent divisions in the landscapes. Examples are the borders of counties, boroughs and parishes.

areas

These are *polygons* that enclose and cover distinct *area features* on the ground, such as buildings, gardens and roads. A polygon is a sequence of vertices assumed to be connected by straight edges, where the first and last vertices are also connected.

Each feature is classified as being of a particular *class*. A line might be of the class road edge, or building outline. To reduce database size, each class may be uniquely assigned a natural number, known as a *feature code*. The feature code, rather than the class name is recorded in the database.



Figure 1.2: Section of line layer from a vector database.

1.2 Creating & Classifying Area Features

Figure 1.2 shows a sample of vector line data for a suburban area. Looking at this projection, we may discern the area features of a landscape, as defined by the line features that bound them. Features such as houses and roads may be readily identified based on their shape. Other areas may be recognized by their proximity to those features, such as the garden around a house. For many other areas, an investigation of the feature codes of the enclosing lines would be necessary for classification.

Modeling the area features of a landscape has many advantages:

- The areas can be feature-coded, have attributes associated with them and be symbolized based on their class when they are displayed. For instance, as seen in Figure 1.3, roads can be shaded in grey, gardens in green and buildings in orange. By referring to an accompanying *legend* such as Figure 1.4, a user can identify the classification of an area.
- A vast number of additional spatial and demographic analyses may be performed on the resultant *polygon* theme, greatly adding to the database's value. For instance one could identify houses that cover an area of $>100\text{m}^2$ and are $\geq 200\text{m}$ from the path of high tension power lines, a retailer could determine the number of homes within the catchment area of a potential development site or all instances of roads dividing water polygons (bridges) could be identified.



Figure 1.3: Polygon layer from a vector database for area shown in Figure 1.2.

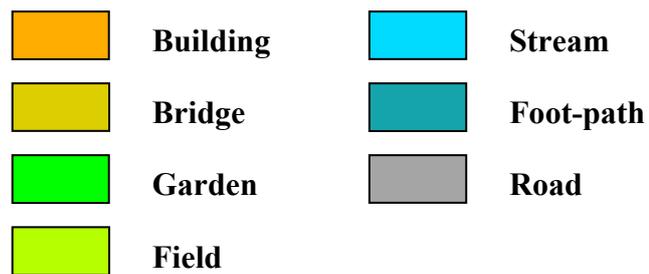


Figure 1.4: Legend for features in Figure 1.3.

The polygon layer may be derived from the line layer by identifying areas that are completely enclosed by line features. The vertices of the lines that form the perimeter become the vertices of the area feature. The *topography* of a polygon is thus defined as a sequence of vertices which are assumed to be connected by straight edges and where the same assumption is made about the first and last vertices.

The feature-coding of area features necessitates the inferring of information that is not present in the existing database. The necessary classification can be achieved through a combination of techniques:

- Inspection of the shape and size of the polygons.
- Examination of *context* within the polygon layer. Areas may be identified by noting the feature codes of surrounding polygons that have already been classified.
- Scrutinising the context of an area feature within other layers, by projecting several layers and their associated attributes together.
- Overlaying corresponding high-resolution aerial photography with the polygon layer, allowing visual identification through colour and texture.
- Surveying the features on the ground (obtaining *ground truth* data visually).

Creating a large-scale topographic database is a huge and expensive task. In most countries, a single national mapping agency is tasked with creating and maintaining a vector database of the entire nation. In the Republic of Ireland, this role is filled by Ordnance Survey Ireland (OSi). A large-scale database shows every building, structure, garden, yard, footpath, road, roadside, traffic island, track, rail-line, field, pond, lake, stream, river and beach. The derivation of a polygon layer from the line layer, as already described, can be automated successfully. Considering the vast number of area features that would be generated nationally, the manual classification of polygons is not feasible. The number of man-hours that would be required makes the task prohibitively expensive. This necessitates the exploration of the automation of the aforementioned classification techniques.

An initial attempt at classification should use information that is already present in the database. Area features may often be classified by reference to the feature codes of the line features from which their topography was constructed. For instance, if a polygon is completely bounded by a building outline, it is probably a building. A feature contained within building outlines, obstructing features (hedge, fence or wall) and path edges is most likely a garden. Paths are normally bounded by path edges and road edges, while roads are usually described solely by road edges.

Once an initial classification has been made, other classification techniques may be applied to classify those features which have not yet been feature-coded, and to check for misclassifications, suggesting alternatives where necessary. We describe the classification of area features by analysis of the feature codes of the neighbouring polygons.

1.3 Context-Based Classification using Structure

Template Matching

The context of a feature is a description of the environment in which it is located. We consider the context of area features within the polygon layer, as described by those polygons which are *adjacent* to the feature in question and the feature codes of those polygons. Figure 1.5 shows a semi-detached house at the centre, with the adjacencies between it and its garden, the house next door and the garden of the house next door highlighted. A corpus of classification *templates* can be built by mining a high-quality *training* data set. Each template describes a particular context that was found in the data along with a record of the number of times a polygon of a particular feature code matched that context. In order to classify a feature, we compare its context to those of the templates. If a match is found, the feature code frequencies that are associated with the template can be used to estimate the probability that the feature is of any given class. This allows the feature to be classified or for the existing classification to be checked.

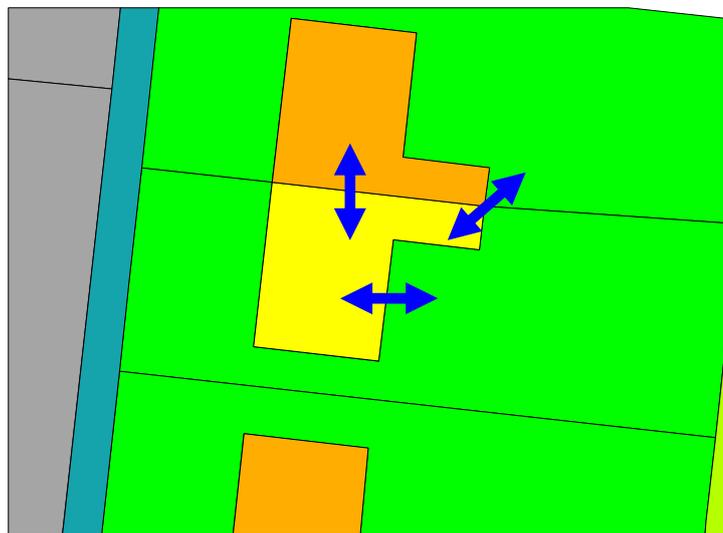


Figure 1.5: Context of a polygon. The adjacencies between the centre area feature and its 3 immediate neighbours are highlighted.

1.4 Aims of Research

The aims of our work were:

1. Implement a tool that classifies polygons in large-scale topographic databases by examining their nearest neighbours.
2. Derive a set of these classifiers from a large corpus of area features.
3. Test them on a separate data set.
4. Evaluate the results and decide if the classification scheme has a practical use.

1.5 Structure of this Dissertation

In chapter 2, we provide an overview of *Geographic Information Systems*, with an emphasis on large-scale topographic databases. The polygon vector data used for this project is described. The third chapter introduces analogies, metaphors and computational analogy models. Chapter 4 describes similarities between geometric analogy problems and topographic data. We detail two context-based classification tools that identify polygons by establishing analogies with templates that summarize previously encountered area features. The fifth chapter describes the pre-processing steps required to prepare data for testing. The implementation of our classifiers and the output of their results are also covered. In chapter 6, the testing of these polygon identification tools is described and the results are presented. Finally, in the seventh chapter, we analyze these results and evaluate our classification scheme. The limitations of our approach are discussed, along with ways of improving on it. We conclude by illustrating other spatial data types that may benefit from our work.

2: Large-Scale Topographic Databases

2.1 Geographic Information Systems

A Geographic Information System (*GIS*) is software that allows a user to collect, store, organise, update, display, query or analyze spatially-referenced information (Longley et al., 2005). Some people consider the hardware platform, the data and the operators to be part of the system. In addition to individual spatial data software applications and installations, the term GIS is frequently used to refer to Geographic Information Systems collectively. Central to many GIS is a representation of certain physical *features* that form the *topography* of a particular landscape. The location and extent of these features are recorded during ground *surveying* or extracted from images produced by *remote sensing* from aircraft or satellites. Topographic features are then modelled within a co-ordinate system that references discrete points on the Earth's surface. A *map projection* is used to map points on the roughly-spherical surface of the Earth to the planar *Cartesian co-ordinate* systems that are commonly used in printed maps and GIS.

2.1.1 The Geographical Co-ordinate System

The shape of the Earth can be described as a sphere which is flattened at the poles and bulges at the equator, due to the centrifugal force of the planet's rotation. This shape may be more accurately characterized as a *spheroid*. Positions on the earth's surface are traditionally measured by a *geographical co-ordinate system* based on degrees of latitude and longitude. This is a form of *polar co-ordinate system* in which angles are measured relative to planes passing through the centre of the earth. Longitude is measured relative to a plane coincident with the axis of rotation of the planet (on which the North and South Poles lie) and a chosen point on the earth's surface (normally Greenwich in Greater London) (Sobel, 1996). Latitude is determined with respect to the plane perpendicular to the axis of rotation; the Equator lies on this plane. Distance is calculated as the length of the curved line along the surface of the spheroid approximation between two points. The geographical co-ordinate system describes features in 3-dimensional space. A point is presumed to lie on the surface of the spheroid approximation, unless its height above sea level is given.

2.1.2 Map Projections and the Cartesian Co-ordinate System

With the exception of the globes common in class-rooms, the earth's surface, or part of it is seldom modelled in a spheroid form. In atlases and other small-scale maps, the geographical co-ordinate system is used, but features are distorted for display on a plane. The mathematical morphing that is used to convert 3-D geographical co-ordinates to a 2-D plane is known as a *map projection*. Many types of map projections have been developed, each of which makes different compromises between distorting the scalar values of *area*, *distance* and *direction* across the map. The large-scale cartographic data described in this dissertation is projected to a plane. The geographical co-ordinates of all features have been converted to the *Cartesian co-ordinate system* (x,y) . Any resulting distortions in the 3 aforementioned scalars are less extreme than they would be in small-scale data and do not impinge this work whatsoever. The use of a Cartesian co-ordinate system simplifies the calculation of the 3 values and many other spatial calculations within a map projection. (Jones 1997, p.61) There are two formats in which Cartesian spatial information is modelled, *raster-based* and *vector-based* data.

2.1.3 Raster Data Model

The raster data model represents spatial information as an array of greyscale or colour *pixels* (a contraction of *picture element*) of regular size. A pixel, also known as a *cell*, is rectangular in shape, normally being a square. Each pixel represents a single value which has been measured / calculated for a corresponding area on the ground. The area represented by a pixel may be defined by the co-ordinates of the bottom left vertex (x_{min},y_{min}) and top right (x_{max},y_{max}) vertex of its *bounding box*, as seen in Figure 2.1. These co-ordinates are not recorded for individual pixels, only the bounding box of the full raster is necessary. As the cells are of a fixed size within a particular raster image, the co-ordinates of a constituent pixel may be easily calculated.

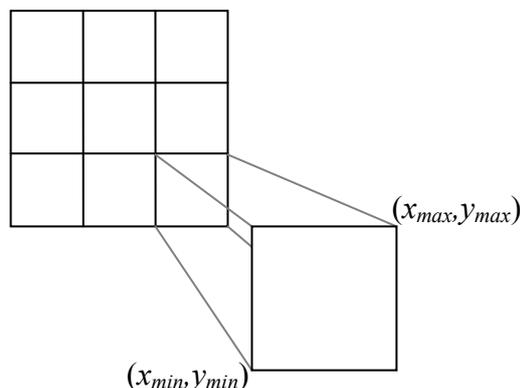


Figure 2.1: The outline of a 3×3 raster image; one pixel is magnified to show its bounding box.

As the raster model emphasises the contents of grid cells in space, it is regarded as *location-based* (Peuquet, 1984). Raster data is designed for use at a specific scale. Converting raster data for display at a smaller scale can be achieved through sampling the original image. Presenting raster data at a larger scale results in the edges between pixels becoming visible, causing the image to appear *blocky* or *pixelated*. *Smoothing* an enlarged image, by averaging adjacent pixel values, may make it more aesthetically pleasing.

All printed maps are raster-based, as are digitised photographs. Many computer graphics formats are *bitmaps* and thus are also raster-based e.g. .bmp .jpg .png .gif. The *Cathode Ray Tube (CRT)* of traditional televisions and computer monitors are described as *raster scan* devices. These and the more modern *LCD (Liquid Crystal Display)* and *TFT LCD (Thin Film Transistor)* flat-panel display screens present images as a raster of pixels.

In addition to aerial photographs, the raster format may be used to show classification over an area. For instance, raster maps are commonly used to show soil or rock types within a landscape. The primary application of raster maps is the representation of topographic surface detail and administrative boundaries. These range from very large-scale town plans to world maps.

2.1.4 Cartographic Symbolisation

Topographic Raster maps represent the outline of features as changes or, in computer vision terms, *edges* in the greyscale or colour pixel values. Features are usually represented in one of three ways:

1. The outlines of the features are shown in a prominent colour, usually black. All other pixels will have a contrasting shade such as white.
2. The area of features is shown in different colours, depending on their class, i.e. they are *symbolized* with a *fill colour* based on their classification.
3. Both techniques above are used together.

2.1.5 Cartographic Generalisation

Three geometrical primitives, points, lines and polygons are used to represent cartographic features. A feature modelled as a polygon on a large-scale map may be represented as a line or point feature at a smaller scale. For instance, the River Liffey might be shown as a polygon on a map of Dublin city centre, and as a line on a map of Ireland. The greater Dublin area may be shown as a polygon on a map of Ireland, but as a point on a world map. *Cartographic Generalisation* is the process of simplifying geometrical features for use at smaller scales. This is necessary to prevent maps from becoming overcrowded and unreadable as scale decreases. It can also be used to automatically generate smaller scale maps from large-scale ones. Techniques used in generalisation include:

- Simplifying lines and polygons by smoothing their outlines
- Merging nearby similar features, such as buildings together
- Translation e.g. moving buildings back from roads to maintain legibility.
- Omitting certain small features entirely
- Replacing area features with line and point features, as already described.

Raster images frequently have place names added as *annotations* (text labels). These are rendered in a particular font and are carefully positioned so as to minimise obscuring of the topographic features or other annotations. The type of annotations used frequently depends on the scale of the image. At a large scale, street names may be shown, while at a small-scale only major roads may be annotated.

2.2 Vector Data Model

The vector data model represents graphics as a collection of Cartesian co-ordinate points, some of which are connected by edges that describe lines and areas. Vector data is the basis of all high-quality scalable graphics e.g. fonts in word processor and desktop publishing documents (PostScript, PDF, Word), Computer-Aided Design drawings, Macromedia's Flash and Shockwave animation formats, W3C's Scalable Vector Graphics format.

Two-dimensional (x,y) vector models are commonly used to represent topographic data in GIS; three dimensions (x,y,z) can be represented by including the height above sea level of points. Metres are frequently used as the units of measure and positions are stored as real numbers of arbitrary precision. Linear features are recorded as sequences of vertices that are assumed to be connected by straight edges. The accuracy of the approximation of real world curved line segments is affected by the frequency with which points along the curve are surveyed and recorded as vertices. More frequent sampling increases accuracy at the expense of data set size. Some GIS can use mathematical functions, such as Bezier curves to more accurately and efficiently represent curves. The vector data we use does not contain curves.

2.2.1 Vector Data Layers

Topographic vector databases organise cartographic features in a set of *layers*. Every feature is associated with a *feature code* which denotes the classification to which it belongs. A typical collection of layers would include the following.

Points

A *point* is a Cartesian co-ordinate with associated attributes. They represent small features whose orientation is not considered important, such as telephone poles and post-boxes.

Symbols

A *symbol* is a special type of point feature which is associated with a graphic which is displayed at a specified angle at its position when the layer is shown on a display or printed. Typical symbols are heritage markers, railway switches and water flow direction indicators.

Lines

A *line* is an ordered set of 2 or more vertices which are assumed to be connected in sequence by straight edges. A vertex is a Cartesian co-ordinate, but unlike a point, it does not have attributes associated with it and exists only as a constituent of a linear feature.

Administrative Boundaries

An *administrative boundary* is a line feature that marks part of the bounds of region(s) defined by humans for administrative purposes. Such regions may include counties, boroughs, electoral wards, and parishes. Boundaries often coincide with the centre-line of a river, stream, road or the edge of a field.

Areas/Polygons

A *polygon* is a set of 1 or more *rings*. A ring is an ordered set of 3 or more vertices which are assumed to be connected in sequence by straight edges, including the first and last vertices. Each ring forms a perimeter of the polygon. By convention, the region to the right of the sequence of vertices that describe a ring denotes part of the inside of the polygon. This means that a ring defined in clockwise order represents a region of the polygon, while an anticlockwise ring defines a hole in a region. Polygons can be of arbitrary complexity, consisting of disconnected regions, nested islands within holes etc. A set of rules enforces the topological correctness of a polygon e.g. a ring may not intersect itself or cross another ring. The polygon layers within the topographic data described here consist of single continuous regions which may contain holes. A polygon with a hole is commonly known as a *doughnut*. The term *area* is used interchangeably with polygon.

2.3 Ordnance Survey Ireland

The object of the research described in this dissertation was the classification of area features in large-scale topographic vector databases. Large-scale vector line data are commercially available from the Irish national mapping agency, Ordnance Survey Ireland (OSi) but are prohibitively expensive. Furthermore, the considerable task of deriving a polygon layer from the line layer would have to be undertaken. As our classification scheme is based on the classes of adjacent area features, the classification of all polygons must be known in advance. As OSi has not developed a polygon layer model to date, we would have to create a list of feature codes to describe all area features that might be derived. The large volumes of *training data* required to develop accurate classifiers and *testing data* to evaluate their performance would need to be of consistently high quality. This necessitates that we have a high level of confidence in the existing classifications of the polygons i.e. that we have *ground truth* data. Ensuring the ground truth of cartographic databases is a multi-disciplinary task suited to large organisations and one which is well beyond the scope of developing our classification scheme.

2.4 Ordnance Survey (Great Britain)

Ordnance Survey (OS) is the national mapping organisation of Britain. Their large-scale vector line data has for many years been available under the *Land-Line* brand-name. In recent years OS developed *The Digital National Framework (DNF)*, which they (Ordnance Survey, 2005) define as:

A nationally consistent geographic referencing framework for Great Britain. Comprising the National Grid and the National Topographic Database that defines each geographical feature as it exists in the real world with a maintained unique reference allocated to each feature. The DNF is not a product; it is the framework on which our future products will be based.

2.4.1 OS MasterMap

Under the DNF, the Land-Line database has been built upon to develop *OS MasterMap (OSM)*, which they dub the “Definitive digital map of Great Britain”. MasterMap offers 4 layers of coverage for the country: Topography, Address, Imagery and ITN (Roads). The fundamental topographic vector data layers of point, line, area etc. are part of the Topography Layer. Some of the key features of MasterMap are as follows (Ordnance Survey, 2005):

2.4.1.1 TOIDS

Each feature in every layer is assigned a *Topographic Identifier (TOID)*. A TOID is a 16-digit natural number which may be used to uniquely identify that object. A feature maintains its TOID throughout its lifetime, unless it is significantly altered, and TOIDS are never reused. TOIDS can be used to associate a user's external data with MasterMap features.

2.4.1.2 A Polygon Layer

In addition to the point, line, boundary, symbology and annotation layers, an area feature layer is included. All polygons are classified according to a fixed list of feature codes and provide continuous non-overlapping coverage (i.e. there are no gaps between polygons and they only intersect along their perimeters) for England, Scotland and Wales.

2.4.1.3 Topological Polygons

MasterMap data is available in topological and non-topological formats. In the non-topological format, polygons are explicitly defined as a sequence of vertices which form their boundary / boundaries. Topological polygons are implicitly defined by references to the features in the line layer that constitute their borders. Topological data, being more structured is more amenable to adjacency analysis and requires less storage space as they involve less data redundancy. Non-topological data can be displayed more quickly and other types of analyses can be performed more efficiently on them.

2.4.1.4 Seamless Coverage

Historically, vector data, including Land-Line had been provided in rectangular sections of fixed size known as *tiles*. Features which spanned multiple tiles were split along the edges of the tiles, causing loss of information to the user. A customer can order MasterMap by a pre-defined area or by an area they define themselves by drawing a selection polygon over a zoomable map of Britain that is presented as part of the online order process. Any feature that intersects the selection polygon is included in the supply.

2.4.1.5 Change-Only Updates

Erosion and human development constantly alter the topography of the landscape. OS continuously update their data holdings based on aerial photography, ground surveys and improvements to positional accuracy. Historically, the update of tile-based vector data required the customer to replace the affected layers of that tile. MasterMap allows for both an initial supply of specified layers and change-only updates for those layers. A

user who wishes to update their data can order change-only updates for the period after the date on which their holdings were last updated. As change-only updates contain far less data than the initial supply they are more likely to be amenable to download from the OS server, rather than delivery on optical media.

2.4.1.6 GML-encoded

There are many commercially available GIS suites, many of which use a proprietary format for managing topographical vector data. This necessitated that mapping agencies such as the OS supplied their data in a wide range of formats. The *Geography Markup Language* (GML) is an XML encoding for the modelling, transport and storage of geographic information, including both the geometry and properties of geographic features. It was developed by the Open Geospatial Consortium (OGC) in consultation with OS amongst others. MasterMap data is only supplied in a compressed GML format. The GIS software publishers provide tools to convert GML to the formats they support and manage change-only updates of data holdings in those formats.

2.4.1.7 SVG Compatibility

Scalable Vector Graphics (SVG) is an XML format for two-dimensional graphics defined by the *World Wide Web Consortium* (W3C). In addition to encoding vector data, images and text are also supported. As GML and SVG are both XML-based, *Extensible Stylesheet Language Transformations* (XSLT) may be used to convert MasterMap data to the SVG format. The Mozilla Foundation has built SVG support directly in to its Firefox web browser. Any browser with an SVG plug-in, such as the free Adobe SVG Viewer can download MasterMap data in a compressed SVG format and render it within a web page using a *stylesheet* to symbolise it. Individual layers and *themes* (such as buildings, land, water and roads) can be selected for display. Maps can be zoomed to arbitrary scales and features can be selected and have their attributes displayed. Graphics can be anti-aliased along their edges, producing very pleasing images on a display of any resolution at any scale, in contrast with the ubiquitous pixelated raster maps. GML and SVG together allow for basic GIS functionality within any web browser.

2.5 Choice of OS MasterMap as Test Case Data

As previously described, the polygon data required for this project was not available from Ordnance Survey Ireland. The British Ordnance Survey's OS MasterMap product includes a polygon layer with the necessary feature codes in place. It is believed that they are the first national mapping agency to develop such a product. The fact that the polygons provide continuous non-overlapping coverage is essential for the adjacency-based classification tool developed. The high quality of the MasterMap polygon layer makes it amenable to a machine learning approach. Importantly, OSGB were prepared to licence the use of their data to Universities for research purposes without fee. There had been a history of collaboration between our research group and the mapping agency. The work described here was supported by OSGB as it constitutes an approach to topographic data quality improvement, the latter being a continuous drive for the agency. Lastly, Ordnance Survey's headquarters and research centre at Southampton and their personnel are within relatively easy reach from Ireland, while a shared language facilitated use of their data and communication.

In the next chapter, we describe the concept of a metaphor and introduce analogy as a key process in learning that uses metaphors to transfer knowledge between domains. Graph structure and isomorphism will be shown to be a key feature of analogies and are at the core of the analogy-inspired polygon classification tools we describe in subsequent chapters.

3: Analogy

3.1 Definition of Analogy

The Oxford English Dictionary (OED, 1989) definition of *analogy* is as follows:

analogy

1. *Math.* Proportion; agreement of ratios.
2. Hence, Due proportion; correspondence or adaptation of one thing to another. *Obs.*
3. Equivalency or likeness of relations; ‘resemblance of things with regard to some circumstances or effects’ (J.); ‘resemblance of relations’ (Whately); a name for the fact, that, the relation borne to any object by some attribute or circumstance, corresponds to the relation existing between another object and some attribute or circumstance pertaining to it. Const. *to, with, between.*
4. *more vaguely,* Agreement between things, similarity.
5. As a figure of speech: The statement of an analogy, a simile or similitude. *Obs.*
6. = [ANALOGUE](#).
7. *Logic.* **a.** Resemblance of relations or attributes forming a ground of reasoning. **b.** The process of reasoning from parallel cases; presumptive reasoning based upon the assumption that if things have some similar attributes, their other attributes will be similar.
8. *Language.* Similarity of formative or constructive processes; imitation of the inflexions, derivatives, or constructions of existing words, in forming inflexions, derivatives, or constructions of other words, without the intervention of the formative steps through which these at first arose.
9. *Nat. Hist.* Resemblance of form or function between organs which are *essentially* different (in different species), as the analogy between the tail of a fish and that of the whale, the wing of a bat and that of a bird, the tendril of the pea and that of the vine.

Dedre Gentner (1983) describes an analogy as a “deep” comparison between systems of objects. A comparison such as “a lion is like a tiger” might be considered superficial as the physical and predatory similarities between both species are well known. In contrast, “Winston Churchill was a British bull-dog” is an analogy. The lack of physical similarity between humans and canines informs the audience that the likeness may be occupational or temperamental. A person who knew nothing of the statesman but had some knowledge of dogs might infer that Churchill was a tenacious fighter. Alternatively, someone familiar with the man’s career, but with little knowledge of dogs, might form a similar opinion of the breed, especially given its name. The name bull-dog may itself be an analogy as the species might have gotten its name due a similarity between its appearance and temperament and that of a bovine bull. A person usually uses an analogy to explain or describe a concept they think may be alien to their audience in terms of one they consider to be more familiar. At times, an analogy may be

reciprocal, as is the case with the Churchill – bull-dog juxtaposition, if the person encountering it is more familiar with the politician than the breed, or somewhat *au fait* with both.

3.2 Metaphor & Analogy

The Oxford English Dictionary (OED, 2001) defines *metaphor* as follows:

metaphor, *n.*
1. A figure of speech in which a name or descriptive word or phrase is transferred to an object or action different from, but analogous to, that to which it is literally applicable; an instance of this, a metaphorical expression. Cf. [METONYMY](#) *n.*, [SIMILE](#) *n.*
2. Something regarded as representative or suggestive of something else, esp. as a material emblem of an abstract quality, condition, notion, etc.; a symbol, a token. Freq. with *for*, *of*.

The term *metaphor* is often used interchangeably with analogy. Metaphors are extremely widespread in everyday speech. So familiar are we with their use, that we may have difficulty identifying them.

- Time is like a line.
 - An event which occurred *earlier* is *behind* us.
 - An event which will occur *later* is *in front of* us.
 - Events occurring *at the same time* are happening *in parallel* or *concurrently*.

This time-line metaphor visualizes time as straight lines in 2-D space. Often, the present is marked at a specific point on a horizontal line, the past extends along the line from the present to the left and the future extends from the present to the right. Events occurring simultaneously at a different location may be shown on a parallel line. People speak of past events as being *behind* them and future events being *in front of* them as, as if time was a line running through the body, which signifies the present.

Metaphors are frequently used to describe scientific principles in terms of more commonly understood behaviour. In 1913, the Danish physicist Niels Bohr introduced the *Bohr Model* to describe the arrangement of particles within an atom. The Rutherford analogy from Gentner (1983), “The atom is like the solar system”, uses this model. This metaphor may be described as follows:

- The atom is like the solar system.
 - A nucleus is like the sun.
 - An electron is like a planet.
 - Electrons orbit a nucleus the way the planets orbit the sun.

The planets of our solar system orbit the sun in elliptical paths on different planes. In diagrams this is often simplified as circular paths on a single plane. A first introduction to atomic structure at early secondary school level may describe electron orbits as concentric circular paths or *shells* in a 2-D plane around a nucleus. The aforementioned metaphor and simplified model of the solar system is used to describe the arrangement. Each *shell* is explained to hold one or more equally-spaced electrons moving in the same direction. The theory of shells being composed of *sub-shells* is introduced at a more advanced level. The shapes of certain sub-shells are described using the metaphors of a ball, and a figure-8 or a dumb-bell. At university level, the Schrödinger wave equation is introduced, from which the probable position of electrons within an atom can be derived. The solar system metaphor is used in secondary schools to teach a simplified version of atomic structure as a diagram of a star system is far easier to understand than a partial differential equation for a wave function.

A young child can suffer a *green-stick fracture* of a bone. This compares the damage to that done to an immature tree branch when it is bent too far. A parliament may be *dissolved*, evoking the moving apart of the members and their re-integration into society!

The historical practice of the rich to be seated at a higher level and to build loftier homes on higher ground than others has contributed to height being a common metaphor for wealth. Society is often viewed as a vertical scale, with wealth increasing towards the top.

Food has been the source of several metaphors that have been used to ascribe the perceived relative worth of a person. Prior to the 20th century, when a wealthy individual entertained guests at home, loaves of bread were often divided among those present in a standard manner. The comfort of visitors was considered paramount, and they were offered the finest part of the loaf, the upper crust. The hosts would enjoy the centre of the bread, while the servants would have the bottom of the loaf, which might be overdone. *The upper crust* came to be a synonym for the aristocracy to denote that they were highly valued members of their society. Dregs are sediments within liquor that settle at the bottom of a vessel and are considered undesirable. The derogatory appellation *the dregs of society* refers to those considered the most worthless, base and corrupt. Cream has always been a highly prized portion of milk. Being less dense than the rest of the milk, it settles on the top, unless the milk has been homogenized. The expression *the cream always rises to the top* is frequently used to suggest that talent or quality comes to the fore over time. All of these metaphors allude to a vertical scale where height is considered desirable. Sometimes this concept is humourously inverted by an observation such as *the scum rises to the top*.

Sporting metaphors are so commonly used that many of them may be considered clichés. They are especially popular with politicians who like to affect the image of an Everyman while couching concepts in terminology that they feel is more comprehensible to the electorate.

- To score an own goal.
- To move the goal-posts.
- To have a good innings.
- To clear a hurdle.
- To be neck and neck.
- First past the post.

It has been argued (O'Donoghue, 2004) that a metaphor highlights existing similar relationships within two distinct domains, while analogy is the process of identifying similarities between a *source domain* and *target domain* and then transferring new information from the source to the target. This view highlights the centrality of learning to the analogy process.

3.3 Analogy in Learning

Analogy underlies many thought processes: learning, language acquisition, classification, induction and creativity. It uses similarity between a new domain and a well-known domain to make the strange seem familiar. Analogy structures our thoughts as we learn or understand new concepts in terms of things we are already familiar with.

3.4 Letter String Analogies

To explain the mechanisms involved in analogies, we begin by looking at *letter string analogies*. These are sequences of alphabetic characters in four groupings, frequently labeled with letters and written in the form:

A : B :: C : D

This is read as “A is to B as C is to D” meaning that when a set of transformations that convert A to B is applied to C, D is generated. A and B represent the source domain of the analogy, while C and D are the target domain. The strings are composed of letters of lower or upper case or a mixture of both. The transformations that the letters may undergo include shifts of a specific number of positions within the alphabet, substitutions, deletions and additions. Shifts are normally limited to one place to the left, or one place to the right, known, respectively, as the *predecessor* or *successor* functions. An example of a letter string analogy where the second letter is replaced by its successor is:

ab : ac :: ef : eg

A *letter string analogy problem* is presented when the final stage D must be determined by the reader. This is represented as:

A : B :: C : ?

These analogy problems are used in intelligence tests, where a number of possible answers may be presented to choose between. If there are three options, this may be shown as:

A : B :: C : 1 ? 2 ? 3

A problem may be considered ambiguous if there is more than one answer generated by transformations using the minimum number of steps e.g.

ac : abc :: dg : deg ? dfg ? defg

The CopyCat algorithm (Hofstadter 1995) solves letter string analogy problems.

3.5 Geometric Analogies

Geometric analogies are arrangements of geometric primitives (circles, squares, triangles etc.) in four groupings. They may be labeled with letters and are written in the same form as letter string analogies:

$$\mathbf{A : B :: C : D}$$

A *geometric analogy problem* is likewise presented when the final stage D must be determined by the reader. Geometric analogy problems are also used in *intelligence quotient (IQ)* tests as it is believed that a high degree of intelligence is needed to solve them. A typical geometric analogy problem is shown in Figure 3.1. The outer square boxes demarcate the boundaries of each set of objects and have no other significance. To facilitate easier grading in IQ tests, several possible answers (stage D) may be presented to choose between.

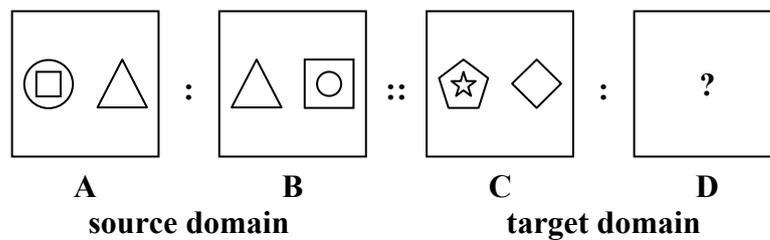


Figure 3.1 A geometric analogy problem without attributes.

Typically, the shapes of the objects, object attributes (colour, size etc.) and the inter-object relations (circle contains square, triangle to right of circle etc.) are significant.

These problems may be solved as follows:

1. Identify the attributes and relations of the objects in A and B.
2. Identify the transformations these values undergo from A to B.
3. Identify the attributes and relations of the objects in C.
4. Apply the transformations to the objects in C to generate the solution, D.

The *Analogy* program (Evans, 1968) takes as input a geometric analogy problem with several possible answers, recognizes the shapes and their physical location with respect to one another and selects the correct answer. It does not handle object attributes (colour, pattern etc.), unlike the *Ludi* model (Bohan and O'Donoghue, 2000). *Ludi* takes as input a predicate representation of the four stages and can generate the required output without being presented with possible answers. An example of a geometric analogy problem with attribute information, which might be solved by *Ludi*, is shown in Figure 3.2.

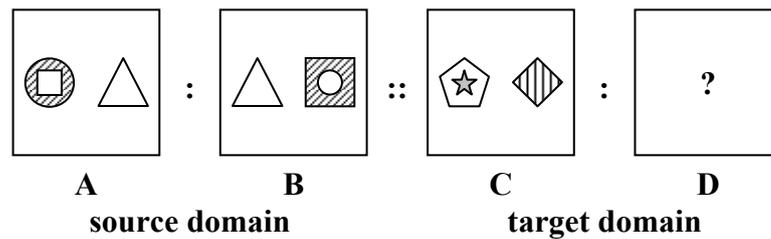


Figure 3.2 A geometric analogy problem with attributes.

The *Ludi* model aims to avoid generating the following trivial solutions:

- D is always identical to B, regardless of A and C.
- There is no solution, because A and C aren't identical and can't be transformed the same way.
- D is identical to C, because only A is transformed.

These answers are considered “simplistic interpretations” of geometric analogies, and they wouldn't be considered satisfactory in an IQ test. Whilst Gentner (1983) dismissed the importance of object attributes (striped, shaded etc.) in favour of emphasizing the structure of the relations between objects, *Ludi* considers both.

Tomai et al. (Tomai et al., 2005) have extended Evans' Analogy algorithm to solve geometric analogies that are presented as sketched inputs. The reasoning is provided by the Structure-Mapping Engine (SME) (Falkenhainer et al., 1989), which implements Gentner's (1983) structure mapping theory.

3.6 Predicate Representation of Analogical Domains

Analogical domains are frequently represented in *predicate calculus* (Waner & Costenoble, 2001). *Predicates* are statements that are known to be true. They are written as *atomic sentences* which are of the form $P(t_1, \dots, t_n)$, where P is the predicate name and t_1, \dots, t_n are one or more *subjects* or *arguments*. Arguments are usually *ordered* or *non-commutative* i.e. in general $P(t_1, t_2) \neq P(t_2, t_1)$. Analogical information is structured in predicate logic as follows:

- *Objects* are the basic entities of analogy and might be tangible items in the real world, actions, events, or concepts. They are usually nouns, but we place no limits on what constitutes an object.
e.g. **egg, floor**
- *Attributes* are 1-place predicates that describe qualities or properties of objects and are often adjectives.
e.g. **fragile(egg)**
- *First-order predicates* describe relations between objects. Frequently, they represent actions and are verbs.
e.g. **hit(egg, floor), break(floor, egg)**
These examples might be one-way relations, i.e. we are specifying that the egg hits the floor and not vice-versa.
- *High-order predicates* describe causal relations between first-order or other high-order predicates.
e.g. **cause(hit, break)**

Predicate form can be more legible than a sentential presentation of the same information. It structures information in syntax similar to that used by the Prolog programming language. Prolog is commonly used in *Artificial Intelligence* research, and is the basis for several computational analogy implementations.

3.7 Analogical Domain Structure & Graph Representation

Some more complex analogies may involve an object in many relations. This structure may be considered as a *graph*, where the objects are the *nodes* and relations are *edges*. “JFK’s White House is like King Arthur’s Camelot” is a well known analogy. It originated in an interview given by the recently widowed Jacqueline Kennedy to Theodore White (1963). A graphic representation of this analogy is shown in Figure 3.3. This is a *directed graph*, as all the edges are ordered (i.e. relations are non-commutative), as denoted by arrow-heads. The objects in both the White House and Camelot domain are arranged so that the relations have the same spatial arrangement in both domains. This makes it easy to see that there is *an isomorphism* between the domains. We can see which objects are analogous to each other: JFK & King Arthur, Oval office & Round Table etc.

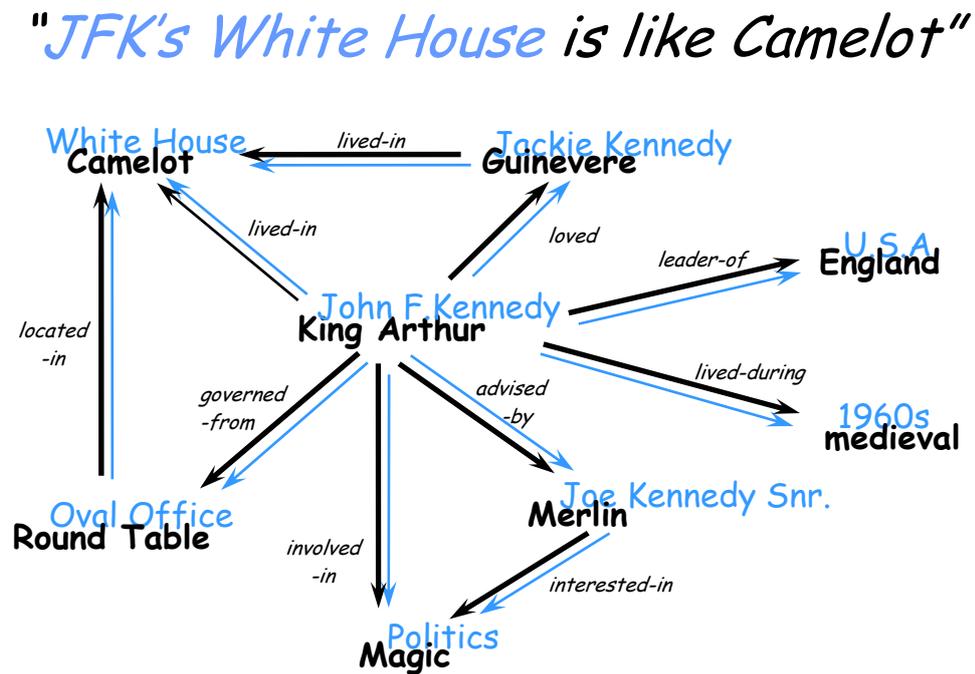


Figure 3.3: Isomorphism of the JFK-King Arthur Analogy. (diagram courtesy of Diarmuid O'Donoghue)

The analogy was developed by White (Sidney, 1994; Wikipedia, 2007c), who drew comparisons between the people, places and events in the lives of U.S. President John F. Kennedy and the legendary King Arthur. It compares his short term in office with the rule of the legendary King Arthur. Both “reigns” are remembered as short-lived golden eras which ended with the tragic death of the central figure. Over the years, this metaphor has been expanded upon by identifying correspondences between both administrations. One of the best known sources of the Arthurian legend today (Wikipedia, 2007a, 2007b) is Sir Thomas Malory’s (2000) *Le Morte D’Arthur*, which was first published in 1485 by William Caxton (who introduced the printing press to England). In Malory’s account, the wizard Merlin advises King Uther Pendragon, who, prior to his death in battle, drives his sword, known as Excalibur, into a boulder. Arthur, unaware that he is a son of Uther, becomes King, when, as foretold by Merlin, he draws the sword from the stone. Excalibur can be viewed as corresponding with the popular vote that mandated JFK’s presidency. King Arthur’s struggles against the expansion of the Roman Empire in Britain (in other texts, he is a British Roman fighting the Saxon invasion) may be seen to mirror the Kennedy presidency’s efforts to arrest the spread of Communism across the Globe. Arthur sends his knights on many quests, the greatest of which is that for the Holy Grail. He believes that drinking from the chalice used by Jesus at the Last Supper will restore his health and the ills that have befallen his kingdom. This echoes JFK’s great desire for his nation to be first to set foot on the moon, an act which would boost national morale and pride, which had been wounded by early Russian victories in the Space Race when they were first to put a satellite in orbit and to put a man in space. Arthur is deceived by his half-sister, Morgause, by whom he produces an heir, Mordred, who Morgause raises to one day usurp his father’s power. Arthur learns of Guinevere’s indiscretions with Lancelot, the greatest knight of the Round Table. This leads to the mortal wounding of the king by Mordred at the Battle of Camlann. What light this may shine on Vice President Johnson, Kennedy family affairs, Lee Harvey Oswald and events surrounding the Texas School Book Depository and the Grassy Knoll is a matter that we leave to the tender ministrations of the conspiracy theorists.

3.8 Structure Mapping

Dedre Gentner (1983) identified *structure* as being the key to unraveling metaphors and analogies. Relations between objects are more important than the objects themselves and their attributes. This allows the identification of *novel metaphors* between apparently unrelated domains. In the “a marathon is like an examination” analogy (O’Donoghue, 2004) seen in Figure 3.4, we can see that the relations within both domains align with each other. It is this isomorphism between domains that underlies the analogy. When the *largest common subgraph* between two domains is identified, this yields a set of source:target pairs of information that is known as a *mapping*. If we compare the mapped relations, we can see that “preparing for” a marathon corresponds to “studying for” an examination. While these relations are not identical, we realize that there is a similarity between both actions. Different *constraints* can be specified to control the types of information that is mapped between domains. In the marathon-examination example the mapped objects are not identical and only some of the relations are the same. In The JFK-King Arthur example of Figure 3.3, none of the objects mapped are identical, but all of the paired relations are equal.

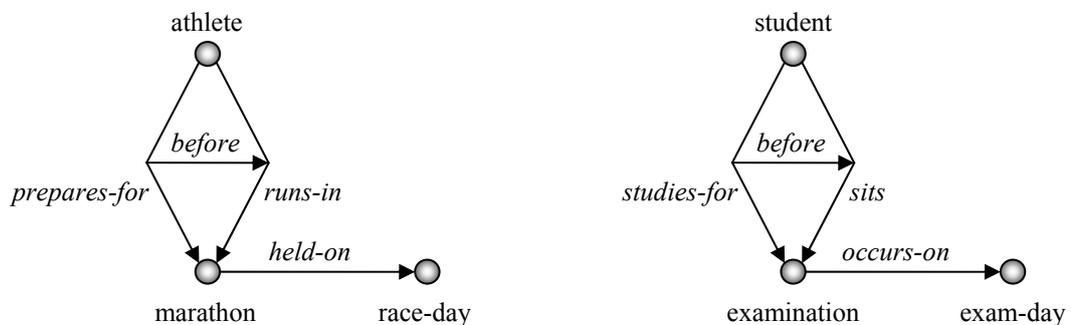


Figure 3.4: The metaphor “A marathon is like an examination”.
(diagram from O’Donoghue (2004, p.5))

An important property of analogical comparisons is *systematicity* (Gentner 1983), meaning that both domains of an analogy must use similar systems of relations in similar ways. The systematicity principle has been the foundation of much research on metaphors and analogy and allows the verification and comparison of different theories. Varying phases in the analogy process have been identified, which emphasize different aspects of the process.

3.9 Phases of Analogy

Keane et al. (1994) identify a five stage model of computational analogy:

Representation, Retrieval, (Structure) Mapping, Validation & Induction.

Using the aforementioned solar system - atom analogy as an example, we describe these 5 phases of analogy.

3.9.1 Representation

The problem domains are described in a uniform way. One might use the predicate calculus description seen in Table 3.1. The solar system domain seen here contains three relations, while the atom domain has two. We will consider the solar system to be the source domain and the atom to be the target domain and try to establish a mapping between both. If successful, we will then attempt to transfer information to the latter.

3.9.2 Retrieval

A knowledge base is searched for possible source domains that might match a selected target. For this example, there is only one possible source domain and all the information in both domains is shown in Table 3.1.

	Source Domain: Solar System	Target Domain: Atom
Objects	sun planet	electron nucleus
Relations	greater-mass(sun, planet) orbit(planet, sun) cause(greater-mass(sun, planet), orbit(planet, sun))	greater-mass(nucleus, electron) orbit(electron, nucleus)

Table 3.1 Predicate representation for solar system - atom domains.

3.9.3 Mapping

A 1-to-1 mapping is generated between the objects of both domains. Initially, the following mapping might be arbitrarily generated:

sun → **electron** **planet** → **nucleus**

This object mapping is represented Figure 3.5.1, where the solar system domain is on the left and the atom domain is on the right. The relative position of the objects between domains denotes the mapping generated.



Figure 3.5.1 First mapping between solar system and atom objects.

3.9.4 Validation

The generated mapping is checked to ensure that it meets specified constraints. In this example, we wish the relations to match. In Figure 3.5.2, relations are represented by pentagons whose direction denotes the order of the arguments e.g. in the target domain, *greater-mass* has *nucleus* as the first argument and *electron* as the second argument, representing the *greater_mass(nucleus, electron)* relation. We find that the *greater_mass* relation does not follow the same direction: the first generated mapping hasn't validated.

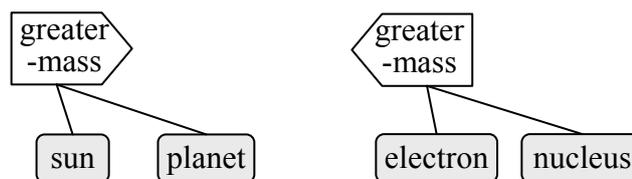


Figure 3.5.2 First mapping fails because relation direction doesn't match.

We return to the previous stage, mapping and generate a second mapping:

sun → **nucleus** **planet** → **electron**

This mapping is shown in Figure 3.5.3



Figure 3.5.3 Second mapping generated.

We now find that both the `greater_mass` and the `orbit` relations match, as seen in Figure 3.5.4. The *cause* relation of the source domain cannot be mapped as it is absent from the target domain. The second generated mapping has validated, so we know it is correct.

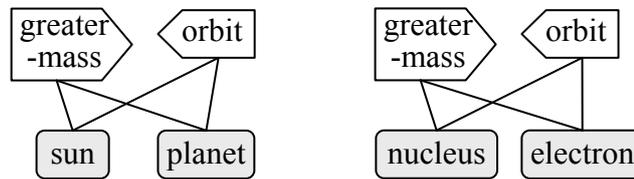


Figure 3.5.4 Both pairs of relations match.

3.9.5 Induction

We transfer information from the source domain to the target domain. The *cause* relation is copied from the solar system domain to the equivalent position in the atom domain, as shown in Figure 3.5.5.

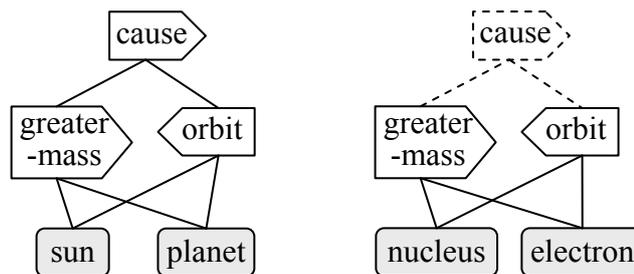


Figure 3.5.5 Additional relation is transferred between domains.

By identifying the largest common subgraph between the solar system and atom domains, we have been able to identify the element that is present in the former and map it to the latter. While there are unmentioned factors that contribute to planetary and atomic orbits, this example shows how analogy can transfer knowledge from one domain to another. If we wished, we could verify the correctness of the new assertion by checking against ground truth.

3.10 Summary

In this chapter we have described the concept of metaphor and introduced analogy as a key process in learning that uses a metaphor to transfer knowledge between two domains. By identifying the largest common subgraph between a source domain and a target domain, we can identify information that is present in the former and map it to the latter. The use of predicate calculus to represent the objects and relations that comprise the domains used in computational analogy was described. Graphs were shown to provide a good visualization of the same information, and make the identification of isomorphism easier. Mapping is described as the central stage in the process of analogy. We used Keane's phases of analogy model to describe the stages in the learning process that are central to our analogy algorithm.

In the next chapter, we return to the large-scale topographic data described in the previous chapter. We describe the similarities between that problem domain and others that have already proven amenable to an analogical approach. We demonstrate that there is structural regularity to much of this data and argue that this structure makes the domain closely analogous to that of geometric analogies. The use of structural isomorphism in topographic data is shown to support a probabilistic approach to classifying map features.

4: Topographical Analogies

4.1 Introduction

We begin this chapter by examining the topological relations that exist between the objects that comprise the domains of geometric analogies. These are compared to the relations occurring between map features in the large-scale topographic databases described in Chapter 2. We show that there are semantic similarities between both sets of relations. Geometric analogies have proven amenable to solution by analogical computation. We argue that the correspondence between the relations seen in both domains, coupled with high degree of structural uniformity exhibited by map data, justifies an analogical approach to the automated classification of features in topographic data. Two important types of adjacency relations found in map data are given special attention. These relations form the basis of two models that describe the neighbourhoods of map features. The identification of isomorphism between these neighbourhood descriptions underpins the identification of analogous features in map data. From this identifier, a probabilistic approach to classifying topographic data by using high quality training data as a source for classification templates is derived.

4.2 Comparison of Geometric Analogies and Topographic Data

We compare the objects, attributes and relations that comprise geometric analogy domains with those found in large-scale topographic vector databases. For brevity, we will refer to the latter as *topographic data* in this section.

4.2.1 Objects

In geometric analogies, the objects are geometric primitives. These may include dots, line features and area features. Topographic data is composed of vector-defined shapes and includes points, lines and polygons.

Object Type	Geometric Analogy	Topographic Data
point feature	•	• telephone pole
line feature	+ ×	 river edge  building boundary
area feature	  	 river  house

Table 4.1: Comparison of object types present in geometric analogies and topographic data.

4.2.2 Attributes

The objects in geometric analogies share several attributes with topographic data features, as shown in Table 4.2. There can be ambiguity in how a visual representation of a geometric analogy is interpreted. The same shape with a different orientation could be considered as a different shape e.g. we could consider an object as having either shape *triangle* and orientation *inverted* or as simply shape *inverted-triangle*. Likewise, a square with a fill colour the same as the background could be considered as an *area feature* with colour *white* or as a *line feature*, which would not have a fill.

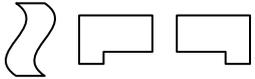
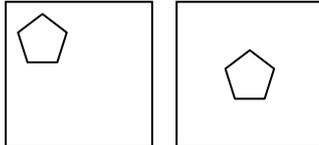
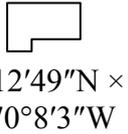
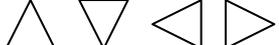
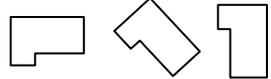
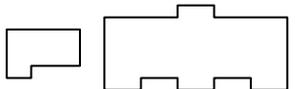
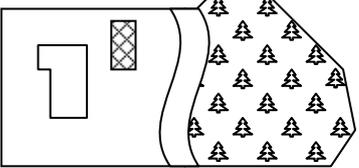
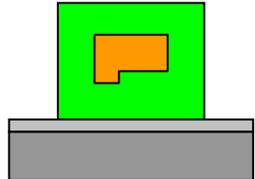
Object Attribute	Geometric Analogy	Topographic Data
Shape		
absolute position		
Orientation		
size/area		
fill pattern		
fill colour		

Table 4.2: Comparison of object attributes present in geometric analogies and topographic data.

4.2.3 Relations

The relations between geometric analogy objects are compared with those occurring between features in topographic data in Table 4.3.

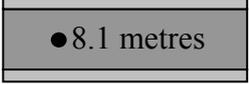
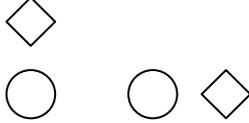
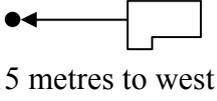
Object Relation Type	Geometric Analogy	Topographic Data
Containment		
relative position		
topology/connectedness		

Table 4.3: Comparison of object relation types present in geometric analogies and topographic data.

The number of possible shapes, attribute values and relations tend to be more limited for geometric analogies because of the requirement that a human user needs to identify these precisely from a graphical representation in order to solve these problems. The spatial precision afforded by the large number of significant digits used in the Cartesian co-ordinate systems that underlie topographic data allows for greater variety in many of these values.

4.3 Applying Geometric Analogy Problems to Topographic Data

In order to solve a geometric analogy problem, one must recognize the objects, object attributes and inter-object relations that comprise these visually-presented tests. For instance, if presented with the problem shown in Figure 4.1, one might describe stage A in predicate logic as being composed of *object1*, *object2* and *object3*, with attributes *circle(object1)*, *pattern(object1,hatched)*, *square(object2)*, and *triangle(object3)* and relations *contains(object1,object2)* and *to-right-of(object3,object1)*. The relations describe the physical structure of the shapes within each stage.

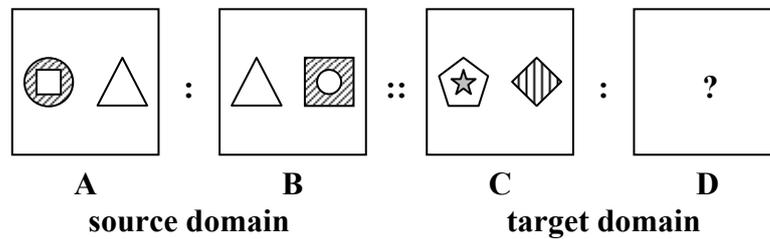


Figure 4.1: A geometric analogy problem using attribute information.

In the first chapter we defined the goal of this thesis as being the automated classification of area features in large-scale topographic vector databases. This requires the inference of the feature code of a polygon from the available feature topography and attributes as described by the various database layers. Classification tools have been developed that use machine vision techniques to identify polygons in a topographic database based on shape, along with *scalar* measures such as area, perimeter and elongation (Keyes and Winstanley, 2001). Aerial photography, geo-referenced to match vector data is also being used to classify area features by inspecting the *texture* of the features (Winstanley and Corcoran, 2005).

As we have seen, there are strong similarities between the geometric analogy domain and topographic data. Geometric analogies are founded on the discrete nature of the attributes of their constituent objects and of the relations between them. We are concerned with classifying area features in topographic vector data. The feature code classes used in such data belongs to a small pre-defined set of descriptions (e.g. building, garden, road etc.) and so may be said to be discrete attributes. We consider again the three types of inter-object relations demonstrated for topographic data in table 4.3. Relative position is a continuous value as the separation between features is measured as a real number of units of distance. Containment is a discrete relation type, but is of limited usefulness as many polygons do not contain other features. An area feature may appear to contain another polygon, when in fact it is a *doughnut* polygon and surrounds the feature which forms the “hole”. A common example of this is a house surrounded by a garden. This leaves topology as the remaining discrete relation type defined for topographic data. Topology may be described in terms of the connectedness or adjacency between features. Feature codes and adjacency are the discrete attributes and relations forming the basis of the topographic feature classification tool described.

4.4 Contextual Analysis of Area Features

It has been observed from the analysis of classified large-scale topographic vector area data that certain classes of polygons tend to neighbour / border on / be adjacent to each other with significant frequency. Salaik (04) analyzed the adjacencies found in the OS MasterMap “Port Talbot” polygon data set containing 5,164 features. Omitting the *add-one smoothing* used by the author, table 4.4 shows some significant feature code adjacency rates from this data. For instance, of the 6,736 instances of adjacencies to features of the “building” class, 3,805 or ~56.5% of these are with polygons of the “garden” class. In other words, for this data set, 56.5% of the neighbours of buildings are gardens.

Feature code		Proportion of adjacencies to class A features by polygons of class B
A	B	
garden	garden	41.07%
garden	building	40.04%
building	garden	56.5%
road	road	41.1%
road	roadside	36.6%

Table 4.4: Selected feature code adjacency rates calculated from Salaik (04).

In addition to considering pairs of adjacent polygons in isolation, one might observe the surrounding *neighbourhood* or *context* of particular area features. There are many definitions of context that one may apply to a polygon in topographic vector data. The topography and attributes of features of any topographic data layer (point, line or polygon) contained within, overlapping or within a specified distance from the boundary or centre of the area feature of interest could be used. One could also consider the relationships between these nearby features and the angles described by these features with respect to each other. We examine the *topological structure* of an area feature and the immediately neighboring polygons and define the context of a polygon as follows:

(polygonal) context

The context of a particular polygon is a description of the classifications of the adjacent polygons and the topology between these polygons.

An adjacency-based description of context is used for a number of reasons:

1. The aforementioned observed regularity in what area features tend to be adjacent to each other
2. The existence of adjacency between two vector-defined features may be readily determined in a GIS. Adjacency is a Boolean value in all cases: either two features are adjacent or they are not. This yields a model of context that is more general and quicker to calculate and compare than many other definitions that could be used.
3. This discrete definition of context is congruous with the clearly defined relationships existing between objects in geometric analogies. As described in the previous chapter, analogy has been successful in solving geometric analogy problems. We examine whether an analogical approach may be applicable to classifying topographic data also.

4.4.1 Types of Adjacency

When considering adjacency between polygons, we distinguish between and define two separate topological relationships:

line-adjacency

Two polygons are line-adjacent if they share a bordering line.

point-adjacency

Two polygons are point-adjacent if they are not line-adjacent but they meet at 1 or more points.

Examples of the two types of adjacency are shown in figure 4.2.

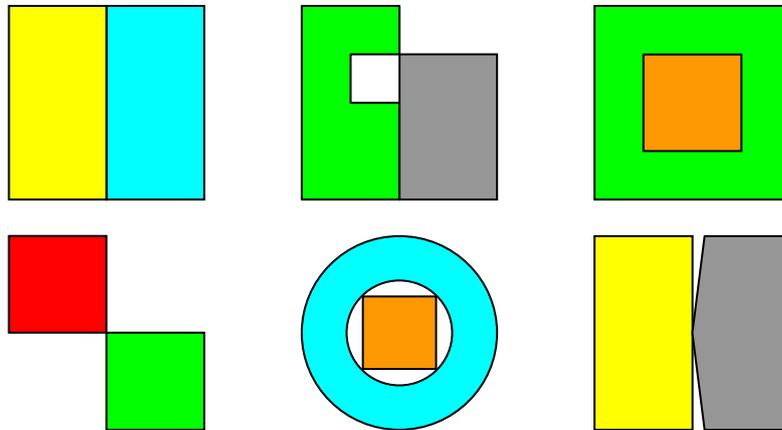


Figure 4.2: The pairs of polygons on the top are line-adjacent, while those on the bottom are point-adjacent.

Point-adjacency is very common between land parcels and buildings. Its treatment as a special case of adjacency allows for a more fine grained description of the context of such polygons. In addition, to avoid the problem of having one unmanageable polygon representing the road network for the whole of Britain, the OS have added polygon closing links at road junctions. These create artificial point-adjacency topology at many junctions, making the road network amenable to context-based classification as well. The distinction between two types of adjacency makes adjacency a trinary value. Any two area features exhibit one of the following relationships:

1. line-adjacency
2. point-adjacency
3. non-adjacency

4.4.2 Graph Representation of Context

The context of a particular area feature may be represented as a *graph*, as demonstrated for the commonly occurring semi-detached house context in figure 4.3. *Nodes* denote polygons and are labeled with feature codes. Identifying numbers are included to aid comprehension. The existence of line-adjacency between two area features is recorded by a continuous line *edge*, while a dashed line *edge* represents a point-adjacency. As normal in graphs, nodes are positioned arbitrarily. The area feature, labeled 1, whose context is being considered is highlighted in both the topographic and graph representations. Its feature code is not recorded in the graph, as a polygon's attributes are not considered a part of its context.

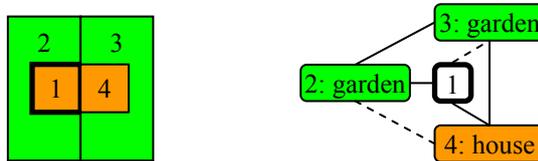


Figure 4.3: The context of the highlighted polygon shown on the left is represented as a graph on the right.

4.5 An Analogical Classification Model for Topographic Data

Having established similarities between the objects, attributes and relations of geometric analogies and topographic data, we describe how an analogical approach, founded on the identification of graph isomorphism between polygonal context descriptions may be used to classify area features in large scale topographic vector data. Figure 4.4 shows the immediate neighbourhoods / contexts of two highlighted semi-detached houses. Both houses are adjacent to one other house and 2 gardens, but the neighbourhoods have differing topographies as corresponding houses and gardens are of varying shape and size. If we examine the contexts of both houses, we find that they share the same topology, as demonstrated by the fact that they generate identical context graphs. On the basis of their isomorphic contexts, we can say that the two highlighted features are analogous to each other.

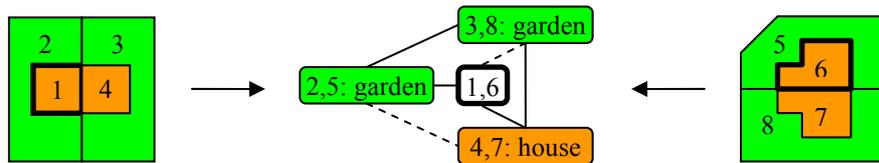


Figure 4.4: Two polygons neighbourhoods of differing topography, may share the same topology, and thus have isomorphic context graphs.

4.5.1 Context Isomorphism

In the previous chapter, we described how analogy is predicated on finding the largest common subgraph between some source domain and a target domain. This facilitates the mapping of corresponding objects and relations from the source to the target. The largest common subgraph between two isomorphic graphs is the entire graph. Thus, identifying isomorphism between two graphs is a subset of the problem of identifying the largest common subgraph. Matching structure between two domains generates a 1-to-1 mapping of objects and relations from the source to the target. Once this is achieved, unmatched objects or relations in the source domain can be mapped to the target domain.

Two graphs must share the same number of nodes, in order to support a 1-to-1 mapping between them and possibly be found to be isomorphic. Given two graphs, A and B , both of n nodes, there are n possible nodes in B that the first node in A can map

to, $n-1$ nodes that the second node in A can map to, $n-2$ possibilities for mapping the third node in A etc. If we map from each node in a fixed ordering of A to a node in the corresponding position of a *permutation* (i.e. an ordering) of B , we generate a set of mapping between A and B . By generating a set of mappings from A to each of the permutations of B , we can generate all possible mappings between both graphs. Overall, the number of permutations of B , and hence mappings between A and B is:

$$(n)(n-1)(n-2)\dots(1) = n!$$

Thus, the computational complexity of determining graph isomorphism is of the order

$$O(n!)$$

As context descriptions have no more than one edge between each pair of nodes, a mapping from the nodes of A to those of B allows the unique mapping of each edge between both graphs also.

There are 2 pre-conditions that must be met in order for there to be a possibility of an isomorphism existing between A and B :

1. Graphs must have the same number of nodes.
2. Graphs must have the same number of edges.

To check whether a given mapping between A and B represents an isomorphism between both graphs, there are two constraints that must be satisfied:

1. Each node in A must map to a node of the same feature code in B .
2. Each pair of nodes in A with an edge between them must map to a pair of nodes in B that share the same edge type.

There is no theoretical upper limit on the number of immediate neighbours that an area feature may have. One particular polygon in OS MasterMap data, representing a footpath surrounding a block of homes in a housing estate, has been observed to be adjacent to over 70 other area features. Given the $O(n!)$ complexity of the operation, checking for isomorphism between two such features could require the generation of up to $70! \approx 1.2 \times 10^{100}$ trial sets of inter-graph mappings, making the problem intractable at present. A method of reducing the search space size, applicable in many cases, will be described in the next chapter.

4.5.2 Context-Based Classification

Presented with the context of a polygon, B , which is either unclassified or whose feature code we wish to verify, we could search a corpus of data for a polygon, A , whose class is known, and whose context is isomorphic with that of B . If a match is found, the context of A could be treated as the source of an analogy and that of B as the target. If

we mapped the feature code of A to B, we would be transferring knowledge between domains. We would be applying the class of A to B on the basis that their contexts are analogous. There may be more than one isomorphism between two contexts. These would differ in what neighbours of A mapped to what neighbours of B. As it is an attribute of A that we seek to transfer to B and both polygons may only map to each other, it is irrelevant which permutation of nodes generates an isomorphism. Once a match is established, we do not need to continue searching for other isomorphisms or even to know if there are any. In the case of two area features whose contexts do match this has the effect of reducing the size of the average search space that is traversed before an isomorphism is established.

4.5.3 Comparison of Geometric and Topographic Analogies

If the classification of some target polygon by a source area feature whose context is isomorphic was depicted as a geometric analogy problem, it might look like figure 4.5. Here, stage A represents the context of the source area feature, stage B shows that polygon's feature code in that context, stage C represents the context of the target area feature and options 1, 2 and 3 are the possible classifications for that polygon within that context that we can choose between. The problem could be read as:

Neighbourhood A implies a house as neighbourhood C implies which of the following?

1. a garden
2. a house
3. a pond

As the context of the source implies a missing house at its centre, and the context of the target is identical, we might reason that the correct answer is option 2, also a house.

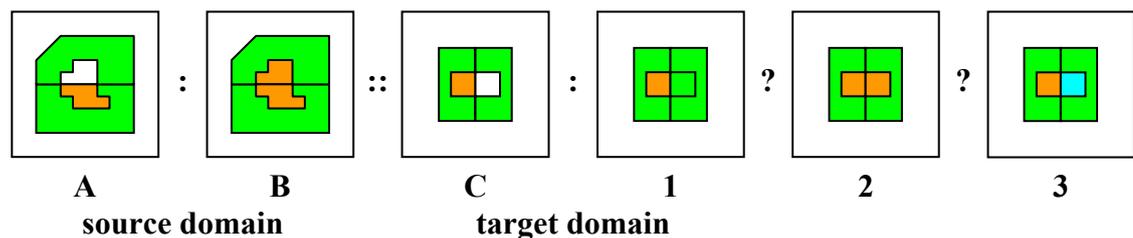


Figure 4.5: The polygon classification process characterised as a geometric analogy problem using object attributes and contextual information.

A geometric analogy problem is predicated on identifying transformations that object attributes and relations undergo within a source domain and applying them to an arrangement of objects in a target domain to generate a solution. Context-based topographical analogies are founded on identifying neighbourhoods of objects, each

having feature code attributes and sharing adjacency relations. Given a target object of unknown class, we identify its context and search a corpus of data for an object with an isomorphic neighbourhood. If a matching source context is found, the feature code of the source is used to classify the target. In geometric analogies, transformations may be applied to many objects and relations in the target, altering it fundamentally. In topographic analogies only the feature code attribute of the target polygon is changed.

4.5.4 Templates and Ambiguity

When a source context is retrieved from a corpus of data and used to classify a given target area feature, we may say that the source context is acting as a *template*. For our purposes, we define this as:

template

An exemplar, an archetype, a typical instance, a guide, a mold, a pattern used in recognition, a context associated with a feature code that suggests a polygon with the same context might share that feature code.

When a corpus of data is searched for a template with which to classify a given area feature, there are four possible outcomes:

1. There is no matching template, in which case our model has no basis for classifying the target.
2. There is a single matching template. We could assume, based on this limited knowledge, that the template is a suitable classifier to use.
3. There are two or more matching templates, all of which indicate the same classification feature code. In this event, the greater the number of matches, the more confident we can be about assigning that class to the target.
4. There are two or more matching templates, but they do not agree on a single feature code. In this instance, our classification scheme may be said to be ambiguous. The feature code applied to the target could vary depending on which template was used.

The randomness introduced by the final possibility clearly would not make for a desirable classification paradigm. Selecting the feature code indicated by a single template and ignoring those suggested by any other templates would be a naïve approach. We describe a more robust scheme which considers all matching templates.

4.5.5 Frequency Templates

As certain contexts, such as those that occur around the highly structured road networks, houses and gardens are very common, we would expect to find many matching neighbourhoods within both a corpus of templates and a sample of polygons requiring classification. It would be very inefficient to repeatedly check for isomorphism between a relatively small number of identical contexts and the entire corpus of candidate sources. Instead, we pre-process the corpus of candidate sources and combine each set of matching templates into one *frequency template*.

frequency template

A frequency template records a context identified in some training set and logs the frequency with which polygons of particular feature codes exhibited that context.

While a template suggests a single class, a frequency template reports frequency counts for each feature code. The creation of frequency templates has the advantages of:

1. Avoiding the inefficiency of repeatedly matching a feature in a testing set against isomorphic templates.
2. Less data redundancy. Rather than keeping a large set of templates in memory when classifying a feature, a smaller set of frequency templates can be used.
3. Addressing the ambiguity inherent when templates do not agree and only one is used.

For instance, a particular frequency template might record a frequency of 5 for “building” and zero for all other feature codes. If this frequency template was retrieved as a match for an unclassified polygon at a later date, we could calculate that a proportion of $5/5$ of the features matching that template in the source were of class building. Alternatively, we could say that the probability that a feature matching that template in the training set is of feature code “building” is 1.0. If we are confident that the topography and feature codes of the training set are representative of those of some testing set, we can say that the *estimated probability* that any feature in the testing set matching that template is likewise “building” is also 1.0. One instance of a situation where we can be almost certain of the structural similarity of two sets of data is if *cross-validation* is used i.e. different sets of features are randomly assigned from a single data set to form training and testing sets. By comparing cross-validation testing results with results obtained from sourcing a testing set from a different data set, we can estimate retrospectively how similar two data sets are.

If a frequency template was to report a frequency of 3 for “garden” and 1 for “building”, the sum of frequencies for that template would be $3+1=4$. The estimated feature code probabilities for features matching that template in a testing set would be $3/4$ or 0.75 for “garden”, $1/4$ or 0.25 for “building” and $0/4$ or 0.0 for all other classes. Based on the sum of frequencies and individual feature code frequencies, a decision can be made on whether to accept the class with the maximum frequency in each case. For instance, a relative frequency of $9/10$ might be considered less reliable than one of $90/100$, despite both representing an estimated probability of 0.9. By generating raw feature code frequencies from our classifier, we give an end user or supervising classification process full statistical information and freedom to decide how to use it. It might be considered desirable, for instance to smooth the feature code frequencies to avoid probabilities of 0.0 for any feature code.

We have established that the computational complexity of Context Structure Matching is $O(n!)$. This makes CSM infeasible for certain features with a large number of neighbours. A model of polygonal context which is less fine-grained than context structures might allow context matching between area features with arbitrarily complex neighbourhoods. Such a model may be based on the notion of *content vectors*.

4.6 Content Vector-Based Classification

A content vector is an array of values that represent information about an object (Marinilli et al., 1999). Each position in a content vector contains a value which describes a specific attribute of the object. To describe the neighbourhood of a polygon, we could use two content vectors, one to record line-adjacencies and one for point-adjacencies. As feature codes comprise a small, finite set, each position in such a content vector could correspond to the number of neighbouring area features of a particular feature code exhibiting that adjacency type. In Figure 4.6, the neighbourhood of a highlighted polygon (labeled 1) is shown on the left as a graph describing its context structure, on the right as a pair of content vectors recording the frequency of occurrence of adjacencies between the area feature under consideration and its neighbours. For simplicity, only four feature codes are used. It can be seen from the line-adjacent content vector, labeled *lnAdj*, that there is one house, one garden and no paths, roads or fields sharing line-adjacency with polygon 1. Similarly, one garden, but no other feature is point-adjacent to the polygon according to the point-adjacent content vector, labeled *ptAdj*. A context structure graph records adjacencies between an area

feature and its neighbours and between those neighbours. The content vectors described only list those between the described polygon and its neighbours.

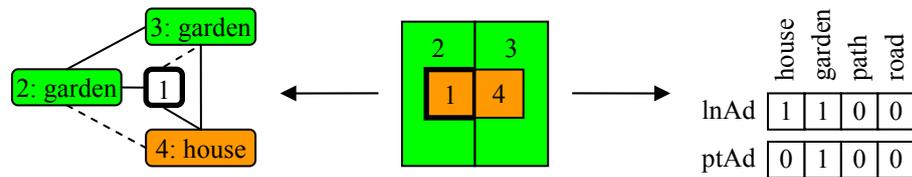


Figure 4.6: A polygon neighbourhood represented as a context structure graph on the left and as a pair of content vectors on the right.

Figure 4.7 shows two similar polygon neighbourhoods, along with their corresponding context structure graphs on the left. As highlighted, the graphs do not match as there is point-adjacency between “our” garden and the neighbour’s house on the left while there is line-adjacency between the corresponding features on the right. However, examining the content vectors for both, seen to the right, we see that these are identical. By disregarding the topological connections between neighbouring features, content vectors provide a more generalized, more coarse-grained model of context than context structures. Checking for matches between two content vectors involves comparing two arrays of numbers each the size of the cardinality of the feature codes used in the source data. As the number of feature codes is finite, the computational complexity of comparing two content vector is constant, $O(1)$, with respect to the number of immediate neighbours of the features. As already described, the complexity of checking for isomorphism between two context structures is $O(n!)$ with respect to the number of immediate neighbours. Thus content vector matching may be attempted in any instance, while context structure matching is intractable in certain cases. As a content vector match is a pre-requisite for a context structure match, the former can be used as a precondition test for the latter, reducing the time to identify non-isomorphic context structures, whilst having negligible impact on the time to determine identical structures.

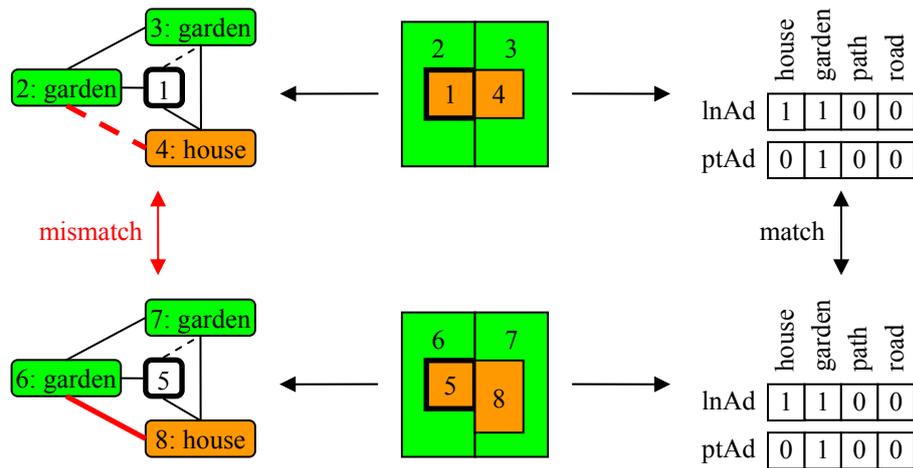


Figure 4.7: Two polygons neighbourhoods with distinct context structure graphs and identical content vectors.

Just as context structure frequency templates may be used as a basis for a probabilistic classification scheme, a separate set of content vector frequency templates can also be used for classification. As context structures are a more fine-grained model of context than content vectors, most content vectors will correspond to contexts that would be represented by many different context structures. A single frequency template of either type records feature code frequencies for all matching neighbourhoods in some source data set. Thus, for the same source data, the set of content vector frequency templates generated can be expected to be significantly less than the number of context structure frequency templates derived. Furthermore, the data structure used to store a content vector contains less information than that used by a context structure, omitting, as they do descriptions of the adjacencies between neighbours. This means that in addition to the computational complexity advantage of content vector matching, the memory size of the frequency templates and the time taken to search and compare them to some target polygon will be significantly less for content vector classification than context structure classification.

4.7 Combining Content Vector and Context Structure Classification

The ability of context structures to represent the adjacencies between neighbouring features makes them a more fine-grained classification tool than content vectors. As several context structures frequency templates will correspond to a single content vector frequency template, we can expect many context structure frequency templates to

record maximum classification probabilities that are greater than that of the content vector frequency templates that encompass them.

The important characteristics of both classification schemes with respect to each other are as follows:

1. Context structures are expected to be a more precise classification tool than content vectors.
2. The graph isomorphism that underlies context structure matching is intractable for large number of neighbours. Content vector matching has constant computational complexity.
3. A content vector frequency template encompasses many context structure frequency templates. If a content vector match is not found, a context structure match cannot be found.

Considering these factors we combine both classifiers as follows:

1. Given a source data set of high quality area features, build a set of content vector frequency templates and a set of context structure frequency templates. Each time a new content vector frequency template is required, a new context structure frequency template must also be created.
2. Given an area feature (in a target data set) which is to be classified, identify its content vector and search for a matching content vector frequency template.
3. If unsuccessful, there is no basis for suggesting a classification.
4. If successful, identify the context structure of the feature. Check if the worst-case computational complexity of establishing isomorphism between two such structures exceeds a predefined limit.
5. If the limit is exceeded, the content vector classification is used.
6. If the complexity is below the limit, search for a corresponding context structure frequency template. If a match is not found, the content vector classification is used.
7. If an isomorphic context structure frequency template is found, use the classification it suggests.
8. Return to stage 2 until classification of all features in the target data set has been attempted.

4.8 Conclusion

In this chapter, we have described an analogical probability-based classification scheme for topographic vector data. In the next chapter, we describe the implementation of the system, looking at the tools used, the pre-processing of the topographic data, the identification of content vectors and context structures, the creation of the frequency templates and their application as a classifier of area features. A number of techniques that allow for the calculation of an upper bound on the computational complexity of context structure matching are described. These allow the use of this more powerful classifier in many cases where we would not attempt its use based on the worst-case scenario of identifying graph isomorphism.

5: Implementation

5.1 Introduction

Thus far, we have introduced large scale topographic vector data and a context-based probabilistic classification scheme to improve the quality of this data. We start this chapter by explaining the GIS and vector data format that we use. The pre-processing steps necessary to prepare the previously described OS MasterMap polygon data for testing purposes is then described. After the extraction of context information from this data is detailed, we explain how we chose our data representation format and implementation platform. Finally, the operation of our template classifiers is described.

5.2 GIS Employed

The choice of Geographic Information System used to process topographic data is restricted by the format of the source data available. Topographic vector data is structured in one of two formats; these are differentiated by the connectedness of features that are composed of more than one vertex, e.g. lines and polygons. Each data type has its advantages and disadvantages for a particular purpose.

5.2.1 Topological Connectedness

Vector data can be categorized as being either *topological* or *non-topological* in nature. In non-topological data, lines and polygons are independent of each other. Each of these features is stored as a data structure that holds the entire ordered sequence of vertices that describe their topography. When two or more lines share an end-point, the common vertex is stored as part of all those lines. When two polygons share a line boundary, the vertices that describe that line are recorded by both area features. By contrast, in topological data, a line starts and ends with a *reference* to a special type of non-cartographic feature called a *node*. A node is a point feature that maintains a set of references to all the line features that it is an end-point of. As a line is composed of an ordered sequence of vertices, one can imagine standing at the start-point, looking down the line and seeing that the line separates a region to the left from a region to the right. A topological polygon is composed of an ordered sequence of reference to line features. Each topological line maintains a reference to an area feature on its left and a reference to an area feature on its right. If a line forms the boundary between 2 polygons, both are

referenced appropriately. If a line bounds an area feature on only one side or not at all, then one or both references are set to *null*.

5.2.2 Topological Data

The topological data format has the advantage of less data redundancy because it does not store identical line topography multiple times. It is better suited to the analysis of topology or connectedness between features. To find if other lines connect to a line feature, we can query its nodes for references to other lines. By checking the left and right polygon references of a line, we can identify the area features which it bounds, if any. To identify the polygons that are adjacent to an area feature, each of the feature's constituent referenced lines are queried for a reference to the polygon on the line's other side. In non-topological data, finding what features are connected to a given feature requires the comparison of the feature's topography/geometry with that of every candidate feature in the data set. Whilst this is acceptable for small databases, performance degrades quickly as data size grows. Topological data is also more suited to maintaining the topographic data holdings of mapping agencies such as the Ordnance Survey. In addition to smaller data size, it makes the continual updating and editing of the data easier and less error-prone. For instance, a line forming the boundary of a house might be part of the boundary of the polygon representing that house and of the garden surrounding it. If a survey found that the house had been extended, the required change to the boundary line would be reflected in the geometry and attributes of the house and garden that referenced it. In a non-topological data holding, the area features bounded by the updated line would have to be identified and have their topography changed independently.

5.2.3 Non-Topological Data

The non-topological data format benefits from faster access to the topography of more complex features and hence quicker display. The topography of a particular feature is stored in a single record (or data structure), unlike topological lines and features, where multiple references to other records must be followed before the full geometry is known. Non-topological data allows faster analysis of features in isolation for this reason.

5.2.4 ESRI ArcView GIS and the Shapefile Format

When this work commenced, there was established, ongoing research into developing classifiers for topographic data within our research group. The topographic data used in testing was large-scale vector data supplied by OSGB under the Digital National Framework. The popular desktop GIS application *ArcView GIS*, published by ESRI Software was available under a site license and was being used within the group for the visualization of topographic data. For this reason, OSGB had provided prototypical DNF data to the research group in the commonly-used ESRI *shapefile* format. Shapefiles are a non-topological vector data format for storing the geometric location and attribute information of geographic features. These were suited to the shape-based analysis of area features for which the data was initially supplied. This OSGB DNF data is believed to have been the only polygonal data available when our work began. The fact that it was provided in the shapefile format dictated the use of ArcView GIS for this project. Automation of the context analysis of shapefile area features is possible through the use of the ArcView's built-in scripting language. Certain operations within ArcView GIS are seen to generate exceptions that reference errors in named files with the *.c extension. This file extension is usually associated with source code files written in the C programming language, leading us to suspect that ArcView GIS is written in some variant of C. No *.c files are included with the installed GIS, and direct access to the data structures that comprise its vector data model is not supported. Interface elements and the topographic feature objects within vector data may be manipulated through the built-in, proprietary scripting language, *Avenue*. Avenue is a verbose, case-insensitive, nominally object-oriented language. It may have been designed for the significant proportion of GIS users with limited programming experience, many of whom have a background in geography. The language (and, arguably, the GIS) is adequate for minor tasks, but doesn't scale well to large data sets. One particular operation, *spatial indexing*, merits special attention as it particularly affects our work.

5.2.5 Optimizing the Performance of Spatial Indexing in ArcView GIS

As already described, a major advantage of topological data is the ease of identifying which features are connected to each other. Determining which features are adjacent to a given object in non-topological data requires the comparison of the geometry of the given object with those of all candidate features. For small data sets, this is not a problem, but the task becomes infeasible as the total number of vertices involved grows. The classic approach that is used to allow this and other spatial searches to scale up involves the *minimum bounding rectangle*.

minimum bounding rectangle

In 2-dimensional vector data, the minimum bounding rectangle of a feature is the smallest rectangle that encloses that feature.

A minimum bounding rectangle (*MBR*) may be fully described by the vertices of two opposite corners, or by one vertex and the displacement to the opposite corner. To check if two features are adjacent to each other, we can calculate the MBR for each feature. If the rectangles are not adjacent, the features they contain cannot be. If they are adjacent, or they intersect, comparison of the vertices of the features can then proceed as normal to determine their relationship. If the MBRs for all features in a data set are calculated in advance, spatial search time may be significantly reduced. ArcView GIS uses a *spatial index* to speed up display and spatial operations (Stellhorn, 2000). Such a spatial index consists of an ordered set of MBRs. It is created and saved in the same directory as the source data the first time a method that utilizes it is called. After the initial overhead of creating the index, it may be used any number of times and will only require updating if the shapefile is edited. However, as the number of features in a layer (and correspondingly the number of MBRs) increases, the performance of ArcView GIS's spatial indexing degrades quickly. This makes processing the largest OS MasterMap data sets available to us infeasible. We overcome this problem by dividing a shapefile into approximately rectangular regions, where each feature that intersects a defined rectangle and is not already included in an adjacent new shapefile, becomes part of a new shapefile associated with that rectangle.

5.3 Preprocessing

Before extracting contextual information from the polygon theme of the OS MasterMap Topography Layer, there are a number of preprocessing steps to be undertaken.

5.3.1 GML to Shapefile Conversion

With the exception of early testing data, all OS MasterMap data is provided in the GML format. We use ESRI's *ARC/GIS Desktop* GIS suite and the separately available add-on data conversion tool MapManager (an "extension" also published by ESRI) to convert this to the shapefile format. MapManager can convert GML to ESRI's *personal geodatabase* format, which can then be *filtered* to generate shapefiles. The process requires the selection of many options within a wizard interface for each stage, but is otherwise automated.

5.3.2 Cartographic Polygon Removal

The vast majority of polygons in OS MasterMap provide continuous non-overlapping coverage of the entire land mass of Britain. These area features are referred to as *topographic polygons*. The OS has also added a series of polygons, which they consider as cartographic constructions to the polygon theme. These *cartographic polygons* belong to one of at least three features classes: slopes, cliffs and pylons. The latter might represent the footprint of an electricity pylon on the ground. OS MasterMap is currently 2-dimensional in nature, with the exception of point features for which height is recorded, e.g. bench marks and spot heights. Cartographic polygons are added-value features that sit on top of a continuous landscape that is represented by topographic polygons. As such they are considered as being above topographic polygons and are normally displayed as such, often using partially-transparent hatched symbology so that the underlying topographic features may be seen. A cartographic polygon covers a region that is already part of one or more of the topographic polygons that provide continuous non-overlapping coverage of Britain. As our classification scheme is based on topographic area features, the relatively small number of cartographic area features is separated prior to adjacency analysis.

5.3.3 Duplicate Feature Removal

The quality of the OS MasterMap polygon layer has improved greatly in the period between its initial development and its launch as a commercial product. Some of the earliest OS MasterMap data, which was made available for testing purposes, is still in use by us as it covers regions for which we do not possess more current data. Whilst the use of better quality data is preferable for testing purposes, the analysis of older data facilitates the evaluation of the dependency of our technique on information quality, and hence its usefulness as a tool to improve such data sets.

Typically, the higher the population density of a region is, the greater the demand is for corresponding cartographic products. As a result, urban areas tend to be surveyed at a larger scale (i.e. at a higher resolution or more accurately) than rural areas. In the UK, large-scale topographic data has traditionally been published at scales of 1:1,250 for urban areas, 1:2,500 for rural areas and 1:10,000 for mountain and moorland areas (Ordnance Survey, 2005). In general, the accuracy of co-ordinates recorded in rural area tends to be lower and fewer vertices are sampled along curving features. In early OS MasterMap polygon coverage, the quality of urban data is noticeably better than that of rural data. This difference might be partly due to the greater degree of quality control the more valuable data is subjected to. In addition, urban data is more frequently updated due to a higher degree of ongoing building, infrastructural and landscape development.

Certain large mainly rural OS MasterMap data sets have been found to contain duplicated topographic area features i.e. features with identical topography. As each region should be contained in only one topographic polygon, all but one of each set of identical features with identical topology must be removed. This is achieved by inspecting the version number and update date information recorded for each feature. Whilst this simplistic approach would be unsatisfactory for updating the master database of a mapping agency, it facilitates the accommodation of otherwise unusable features in our classification scheme. As our technique involves the analysis of the immediate neighbours of each polygon, invalid context information would otherwise be derived for both a feature which has been duplicated, and its neighbours.

5.3.4 Perimeter Polygon Removal

Our classification scheme is predicated on the identification of all the immediate neighbouring area features of individual polygons. The OS MasterMap topographic polygon theme provides continuous non-overlapping coverage for the whole of Britain, out to its foreshores i.e. the shore area between the high- and low-water marks. With the exception of the foreshore areas, it should be possible to determine the context of all polygons within the national OS MasterMap database. If required, the presumption could be made that the foreshore features border on a notional “sea” area feature. Only a fraction of the national database is available to us. The mostly disjoint data sets we do have the use of are composed of all the features from the OS MasterMap topographic layer that are either wholly or partially contained within individual rectangular regions. Without reference to external information, the context of area features on the edge of a shapefile cannot be known. All the polygons on the perimeter of such a region are adjacent to one or more area features that are not included in the data set. We need to identify these as their complete context cannot be determined from just the containing data set.

Our solution to this problem, involves the use of a topographic vector feature, known in ArcView GIS as a *polyline*. Like a polygon, a polyline is composed of one or more enclosed rings, each of which is comprised of a sequence of vertices. Unlike a polygon, a polyline does not have an area, and so it can represent the outline of a polygon. The identification of area features on the perimeter of a small data set is demonstrated in Figure 5.1. The stages numbered in the diagram are as follows:

1. All topographic area features are *merged* into a single new polygon that covers the entire region.
2. The merged polygon is converted into a *polyline* object, yielding the outline/perimeter of the data set.
3. The features and the polyline are projected together/overlaid to allow spatial comparison.
4. Each polygon is checked for intersection with the polyline. If there is an intersection, then that feature is either line- or point-adjacent to the perimeter of the data set and is marked/highlighted
5. The polygons on the perimeter of the data set can then be separated from the internal features.

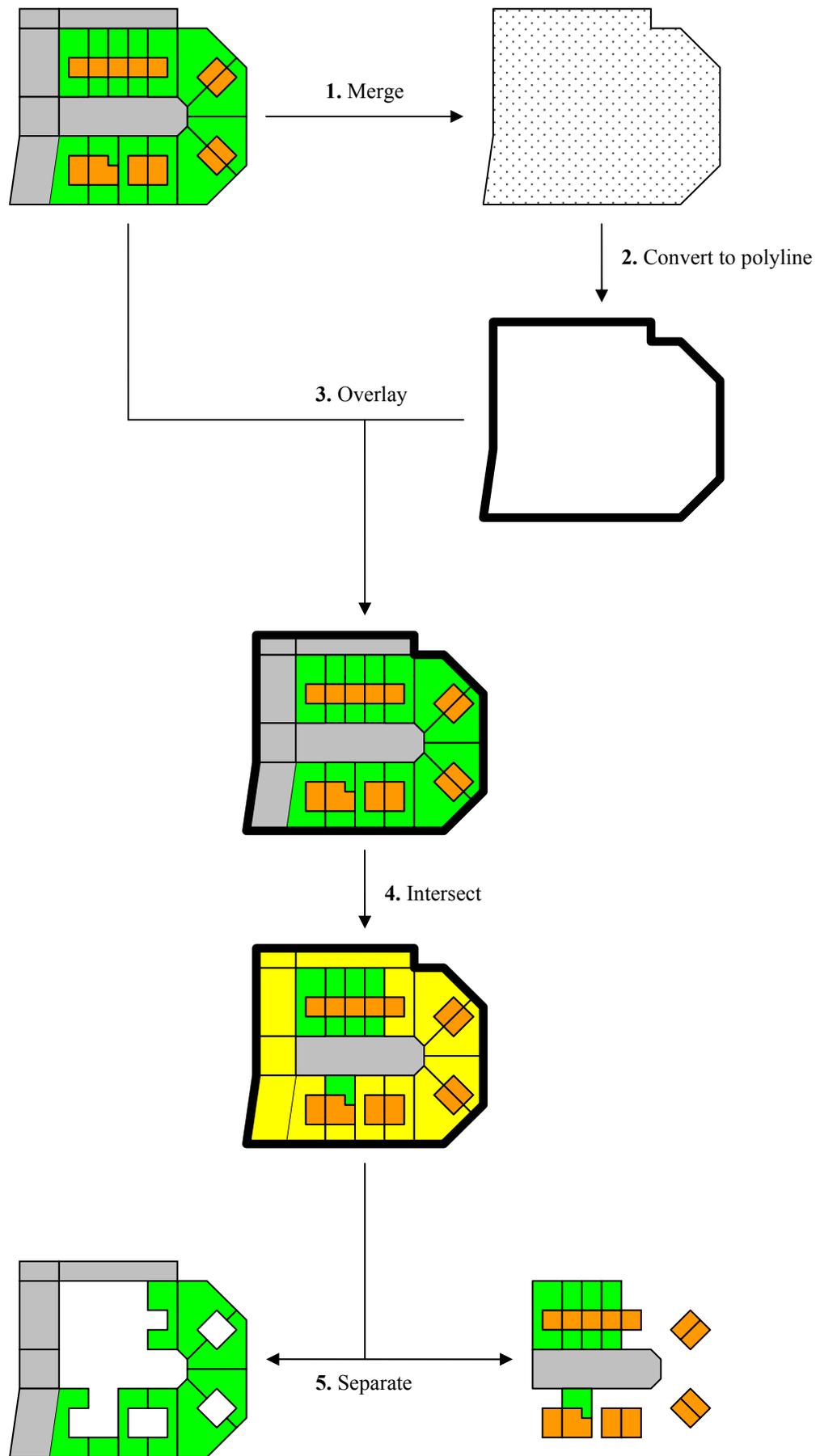


Figure 5.1: Stages in separation of polygons on perimeter of a data set.

5.4 Context Extraction

Having completed the preprocessing steps to identify features that are unsuitable for context analysis, the context analysis of the remaining topographic area features may proceed. We first consider the data structure used to store this context information.

5.4.1 Data Structure Representation of Polygonal Context

As already described, in non-topographic data, the identification of which features are adjacent to a particular object requires the checking of whether each other feature in the data set intersects that object. Even with our optimizations to spatial indexing within ArcView GIS, the identification of adjacency, like all other spatial operations in non-topographic data, is computationally expensive. The model of context of an area feature that we have chosen involves the identification of the adjacent area features of each polygon and the determination of whether adjacency exists between each pair of these neighbours. When analyzing the context of all features in a theme, this results in the checking of adjacency between each pair of features at least twice, and usually far more frequently. This is highly inefficient if polygon intersection is checked each of these times, especially as data set sizes grow. An alternative is to check for adjacency between each pair of features once and store the result for future reference in a *look-up table*.

5.4.1.1 The Adjacency Matrix

Our *adjacency matrix* is a two-dimensional look-up table, where the axes correspond to an ordered sequence of feature identifiers. Each cell in the table records whether the features referenced on the axes are mutually line-adjacent, point-adjacent or disjoint (i.e. non-adjacent). OS MasterMap has a single set of identifiers, sixteen-digit natural numbers known as TOIDs, each of which may represent a topographic feature of any class. Only a small fraction of these 10^{16} numbers represent features of the polygon class, and these are not restricted to a specific range. It would be infeasible to use TOIDs directly as column and row references as the resulting table could potentially be of dimensions $10^{16} \times 10^{16}$. Even if it was possible to store the resulting $\sim 10^{32}$ elements, only a fraction of them would be used. It is highly unlikely that all the features in a particular data set layer would possess a continuous sequence of TOIDs, making the use of an offset of these identifiers (e.g. subtracting the lowest TOID in a data set from all identifier values) impracticable as the potential upper limit of the required table size would still be $\sim 10^{32}$. To eliminate redundant rows or columns, identifiers may be sorted

and mapped in sequence to the column and row numbers. A simple example featuring a data set comprised of five features with single-digit identifiers and a corresponding adjacency matrix is shown in Figure 5.2.

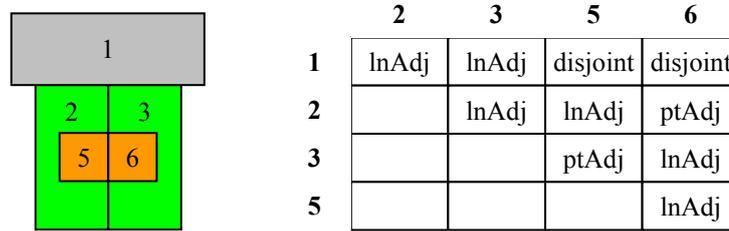


Figure 5.2: A 5-feature data set and the corresponding adjacency matrix.

Here, the rows and columns are labeled with the identifiers of the features they correspond to. It can be seen that there is no row or column for a feature with identifier 4 because such an object is not part of the data set. Because adjacency is a commutative relation, many cells need not be used because they would hold redundant information e.g. if 2 and 3 are line-adjacent, 3 and 2 must be line adjacent. Approximately half of the table is unneeded if each pair of identifiers is always accessed in the same order. For the matrix shown, the smaller of each pair of identifiers accesses the rows, leaving the larger number to reference the columns. In addition to the cells that are unneeded due of commutativity, there is a superfluous series of cells running diagonally down and to the right of the table because adjacency between any object and itself is meaningless and should never be queried. In the same diagram, feature 1 has been excluded from the columns because all its adjacencies are recorded along a row. Inversely, feature 6 is excluded from the rows as all its neighbours are recorded down a column. For these reasons, when a look-up table cross-references a set of n objects, the required table size is as shown in Equation 5.1.

5.1

$$(n-1) \times (n-1) = n^2 - 2n + 1$$

Utilizing the redundancy arising from the commutative adjacency relation, the relation between each feature and those with a larger TOID need only be recorded, meaning that the minimum number of cells used would be as shown in Equation 5.2.

5.2

$$(n-1) + (n-2) + \dots + 2 + 1 = n \times \frac{n-1}{2} = \frac{n^2 - n}{2}$$

The cells in a look-up table may be rearranged to minimize the number of redundant elements. This can be achieved by remapping references in certain instances, but it is questionable whether the computational overhead is worth the saving in space, which rises to nearly half as n grows. Figure 5.3 shows the same 4×4 adjacency matrix seen in Figure 5.2, then that table rearranged as a 2×5 compacted adjacency matrix, where the shaded column references and cells, along with part of the hatched row references, have been remapped. In the reduced table, the clear cells are accessed as normal by the clear row and column references, while the shaded ones have had their row and column references remapped.

	2	3	5	6
1	lnAdj	lnAdj	disjoint	disjoint
2		lnAdj	lnAdj	ptAdj
			ptAdj	lnAdj
5				lnAdj

	5	6	
	disjoint	disjoint	1
	lnAdj	ptAdj	2
	ptAdj	lnAdj	3
2	lnAdj	lnAdj	5
1	lnAdj	lnAdj	
	3	2	

Figure 5.3: An adjacency matrix rearranged as a compacted adjacency matrix.

For an adjacency matrix, a binary search algorithm may be used to identify the row or column corresponding to a particular TOID, so that the cost of look-up would be $O(\log_2 N)^2$. The access time would be expected to increase slightly for a compacted adjacency matrix, due to the added mapping cost. One of the largest data sets available to us is composed of over 240,000 cartographic polygons. By Equation 5.2, the corresponding compacted adjacency matrix would be of size 28,799,880,000. If two bits are used to store the three possible adjacency relations, the table would be ~6.7 GB in size. The exponential growth in look-up table size as the number of features in a dataset rises makes the adjacency matrix approach infeasible in general. This leads us to an alternative approach, which is based on recording where an adjacency exists, but not vast majority of cases of non-adjacency.

5.4.1.2 Prolog Implementation

Prolog is a well-known declarative programming language. It was originally an interpreted language, but some implementations allow for far more efficient compiled modes. The essential structure of a Prolog program is that of a set of known facts, represented as predicates, along with a set of rules which are often highly recursive in nature, and a single top level query. All of these comprise the *database* of knowledge

possessed by the system during execution. Using a built-in backtracking search mechanism, the interpreter or compiler attempts to answer the query by reference to the presented facts and rules. It will search exhaustively until all possibilities are explored without success, or if a solution is found, give the option for the search to continue for other solutions. Libraries of predicates allow for a range of data input and output formats. Prolog is popular in Artificial Intelligence (AI) research as it is suited to the representation of knowledge and deductive reasoning strategies that typify this area. It often allows for the implementation of algorithms as more concise programs than would be possible in the imperative or object-oriented programming paradigms. Within ArcView GIS, Avenue is a severely constrained imperative and quasi-object-oriented language with rudimentary data structure support. Analogy is one area of AI research where Prolog is commonly used. As this work arose from a desire to apply an analogical approach to topographic feature classification, it was decided to use Avenue to extract the required context information and export it to a text file which is used as input to a classification program implemented in Prolog. This leads us to seek a predicate logic representation of topographical feature context.

5.4.1.3 Atomic Predicate Representation of Context

The geometric analogy algorithm LUDI (Bohan & O'Donoghue 2000), describes each object attribute and inter-object relation as a predicate assertion. An example of a geometrical analogy domain and the corresponding predicate logic description is given in Figure 5.4.

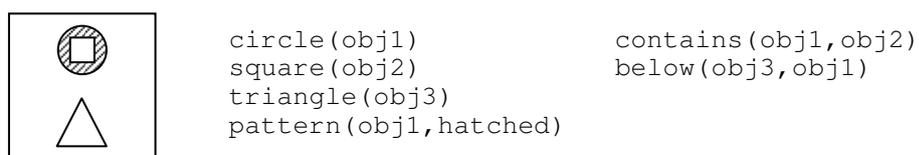


Figure 5.4: A geometric analogy domain and its atomic predicate logic representation.

In the predicate description shown, the geometric objects are arbitrarily referenced as obj1 etc. and four attributes are listed, followed by two relations. This predicate representation may be considered to be *atomic* as each predicate records a single piece of information, whereas a *complex* or nested predicate could describe an entire domain. An atomic predicate representation of context in topographic data was initially considered. A sample topographic data-set, along with its feature codes and topology represented as a graph and then in predicate logic is shown in Figure 5.5.

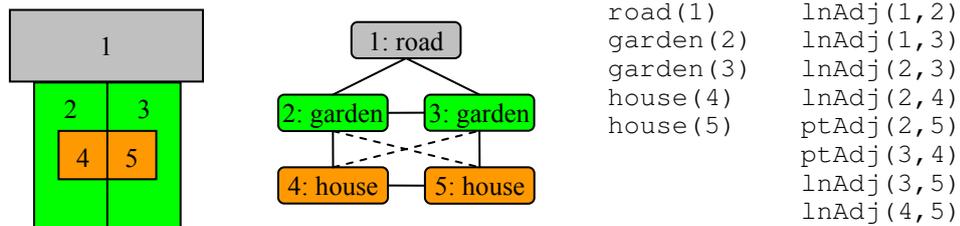


Figure 5.5: A 5-feature data set, its topological graph representation and its atomic predicate logic representation.

In this example, it can be seen that the commutative nature of adjacency has again been used to avoid duplicating information e.g. the line-adjacency between 1 and 2 is recorded as a predicate, but that between 2 and 1 is not explicitly stated. This predicate approach allows for a far more concise description of adjacency than a compacted adjacency matrix, because it does not record the non-adjacencies between disjoint features. To determine the context of a feature from an atomic predicate representation, we would

1. Check all adjacency predicates to identify the neighbours of the feature.
2. Check the feature code predicates until the class of each neighbour is known.
3. Check all adjacency predicates to determine the relation between each pair of neighbours.

In a Prolog implementation, this would require searching the entire database many times. As data set size grows, this would become infeasible.

5.4.1.4 Complex Predicate Representation of Context

The compacted adjacency matrix becomes impractical due to data size, while atomic predicates become unwieldy due to retrieval times. We take a complex predicate approach, which involves an element of data redundancy which avoids excessive retrieval times, while keeping data size proportional to the number of features. A single predicate records the full context of a feature. Figure 5.6 shows the neighbourhood of a highlighted area feature, the context graph of that polygon and a sample Prolog predicate that also describes the full context of the feature. In our implementation, actual sixteen-digit TOIDs identify each feature and the five-digit feature codes are used rather than the verbal approximations (house, garden etc.) that we utilize here for clarity.

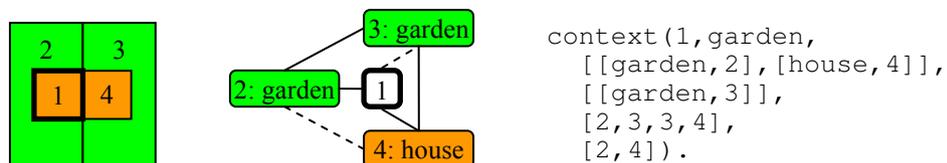


Figure 5.6: A polygon neighbourhood, its context graph and its context predicate.

Any valid Prolog identifier that was not already in use could have been used in place of context as the predicate name. The predicate is comprised of 6 arguments, which in sequence are:

1. **TOID** of the feature whose context is described.
2. **Feature code** of that polygon. This is not part of that feature's context, but it is required in order to develop our statistical classification templates from training data and to evaluate these by reference to ground truth in testing data.
3. **Line-adjacent neighbours** list. A list of lists detailing the line-adjacent neighbours of the feature. The first element of each sub-list is a feature code and is followed by the TOIDs of all corresponding polygons that are line-adjacent to the feature. Sub-lists are ordered by ascending TOID.
4. **Point-adjacent neighbours** list which is structured in the same way.
5. **Line-adjacencies between neighbours** described as a flat list of TOIDs of the features concerned. Each pair of TOIDs in sequence denotes the existence of line-adjacency between those features.
6. **Point-adjacencies between neighbours** structured the same way.

As a content vector is identical to a context structure, but without inter-neighbour adjacencies described, the corresponding arguments 5 and 6 of the context predicate are disregarded during content vector matching. This avoids the need to create, store and retrieve a separate set of predicates as training data or testing data for the content vector classification process.

5.5 Classification Template Construction

A context structure matching template (*csmTemplate* predicate) is very similar to a context predicate, but instead of recording the feature code of a single corresponding area feature, it records the number of polygons of each feature code that match its structure in some training data set. An *association list* is a SICStus Prolog data structure that implements a finite mapping as an *AVL tree* i.e. a binary tree that is subject to the Adelson-Velskii-Landis balance criterion:

A tree is balanced iff for every node the heights of its two subtrees differ by at most 1.

For an AVL tree, look-up, insertion and deletion are all $O(\log n)$ operations in the worst case (Wirth 1976 cited Intelligent Systems Laboratory 2001). Within each template, we use an association list to map from each feature code encountered to an incremental count of the occurrence of correspondingly classified polygons exhibiting that context. Only the feature codes of features found to match a template are included in the association list. This has the advantages of reducing storage space while allowing the code to be used with any set of feature codes. As it is not possible to change a data structure / declared predicate in Prolog, templates are declared as *dynamic predicates*. This allows us to use the *assert* predicate to add a template to the database, and utilize the *retract* predicate to erase a template so that it may be reasserted with an updated association list. A content vector matching template (*cvmTemplate* predicate) also uses an association list to record feature code frequencies in the same manner, but omits the lists that describe adjacencies between neighbours. Context structure templates and content vector templates are derived from a training data set of contexts as follows:

```
For each context structure
  Search cvmTemplates for a match
  If unsuccessful
    Assert new cvmTemplate
    Assert new csmTemplate      // 1
  Else
    Update matched cvmTemplate FC frequencies
    If CSM complexity < limit  // 2
      Search csmTemplates for a match
      If unsuccessful
        Assert new csmTemplate // 3
      Else
        Update matched csmTemplate FC frequencies
```

A number of lines in the preceding pseudo-code are labeled for clarification as follows:

1. As a context structure is a more fine-grained form of a content vector, we know that if we don't find a matching cvmTemplate, there cannot be a matching csmTemplate.
2. The user can specify the maximum theoretical search space that is to be traversed. Above this limit, CSM is not attempted.
3. If there is a matching cvmTemplate, there may be one or more corresponding csmTemplates, one of which might be isomorphic to the context structure we are trying to match. Failing that, a new csmTemplate must be asserted.

Having derived a set of classification templates from some training data set, we can then proceed to classify features within a testing data set. This is accomplished by determining the most frequently occurring class for a particular template. In the next chapter, we describe the training and testing of our classifiers, and present the results.

6: Results

6.1 Introduction

Having described our context-based polygon classification scheme in earlier chapters, we now present the results of testing the classifiers. The first stage of this is the derivation of statistical template classifiers from a large corpus of high quality polygon data. These templates are then utilized to classify as many area features as possible in a separate high quality testing data set. Classification is performed separately by both content vector and context structure templates and the results are combined to improve the classification rate. By checking the classifications suggested by these techniques against the ground truth of the classes recorded for each feature in the testing data set, we evaluate the accuracy of our classifiers.

6.2 Training and Testing Data Sets Used

As previously described, large-scale polygon vector line data from Ordnance Survey's OS MasterMap Topographic Layer product was used for building classifiers and testing them for this project. The training data set used covers a section of the large town of Basingstoke in Hampshire, England and part of the surrounding countryside. The data set includes suburban, industrial, urban and rural areas. It was provided through the OS website as sample test data, split into four compressed GML files with names as follows:

6745-SU5951-2c4.gz 6745-SU5953-2c2.gz
6745-SU6151-2c1.gz 6745-SU6153-2c3.gz

The data set comprises of 37,408 topographic area features. 36,847 of these are internal polygons, and, as such have identifiable contexts. As this data set covers approximately the northwestern quadrant of Basingstoke, we refer to it as *basNW* for brevity.

The testing data set utilized covers the southern part of the town of Port Talbot in Glamorgan, Wales and includes suburban, industrial and rural areas. It was made available through the OS website as a single compressed GML file:

10254-SS7886-5i1.gz

The data set consists of 5,274 topographic polygons, 5,070 of which are internal, and we label it as *port*.

Table 6.1 shows the breakdown of both data set corpora by feature code and the corresponding class descriptions. Feature codes are listed in descending order of their frequency in the port data set. The class *multiple surface land* refers to the grounds surrounding homes, and may be combinations of planted and paved areas. *General surface* is applied to similar areas surrounding non-domestic buildings and to agricultural land. *Roadside* can be grass verges or foot-paths. *Natural environment* describes wilderness. *General surface step* describes areas of steps adjacent to general surface land parcels. *Railway* is the generally enclosed area of land beneath and surrounding train tracks. The class *structure* is applied to significant manmade objects that are not considered buildings. This includes footbridges, but road bridges are classified as areas of *road or track* and *roadside*. *Upper level of communication* is defined (Ordnance Survey, 2005) as

Upper level of through public communication, for example, in multilevel shopping centres.

The larger basNW is used for training as it can be expected to yield a wider range of classification templates. In addition, the higher frequencies of occurrence for the more common contexts results in templates with more accurate feature code probabilities. Figures 6.1 and 6.2 show a graphical representation of the class composition of the training and testing corpora. It can be seen that the proportion of buildings is quite similar across both, while port has less multiple surface land and more general surface, reflecting a lower concentration of suburban homes. Also of note, basNW has smaller proportions of road or track, natural environment and *inland water*, but a greater concentration of *paths*. The differences in the other classes are less significant because of their relatively small numbers. Overall, given the data sets available to us, the basNW and port corpora are good matches for the most common classes seen in OS MasterMap urban/suburban topographic polygon data, i.e. building, multiple surface land, general surface, road or track and roadside.

		class													
		all classes	building	multiple surface land	general surface	road or track	roadside	natural environment	inland water	general surface step	railway	structure	path	glasshouse	upper level of communication
feature code		1002 1	1005 3	1005 6	1017 2	1018 3	1011 1	1008 9	1005 4	1016 7	1018 5	1012 3	1006 2	1018 7	
basNW	internal polygon count	3684 7	1524 9	1361 4	3630	1395	1835	225	16	297	8	26	525	18	9
	% of data set	100	41.38	36.95	9.85	3.79	4.98	0.61	0.04	0.81	0.02	0.07	1.42	0.05	0.02
port	internal polygon count	5070	2138	1284	713	317	296	130	85	39	25	23	17	3	0
	% of data set	100	42.17	25.33	14.06	6.25	5.84	2.56	1.68	0.77	0.49	0.45	0.34	0.06	0

Table 6.1: Composition of internal polygons in basNW & port data sets by class.

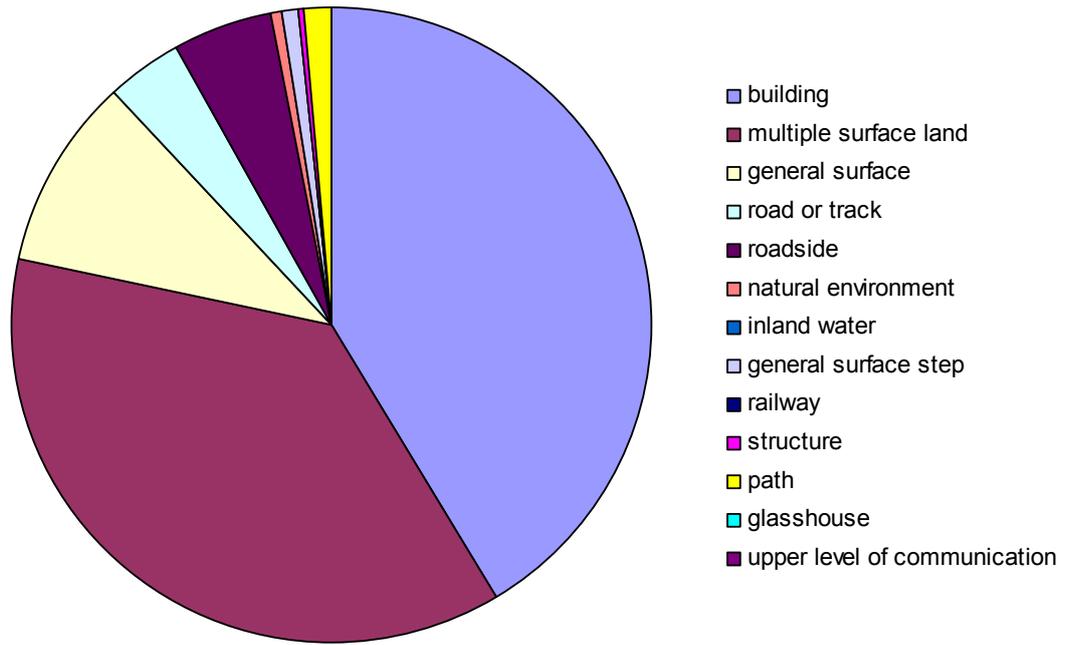


Figure 6.1: Composition of internal polygons in basNW data set by class.

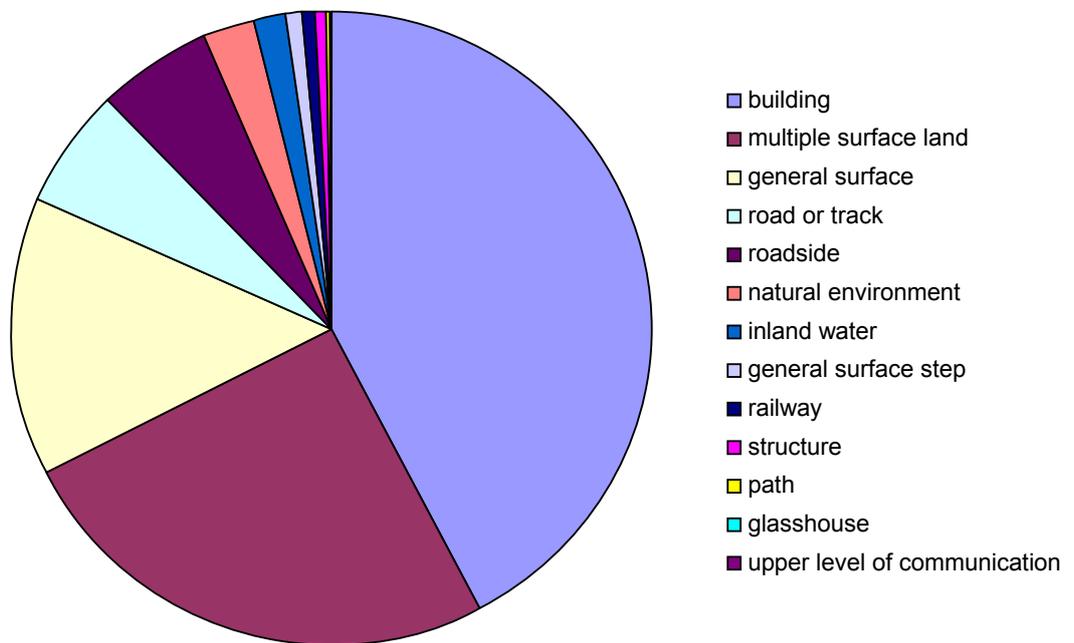


Figure 6.2: Composition of internal polygons in port data set by class.

6.2 Probabilistic classification via CVM and CSM

Each content vector template and context structure template records the frequencies with which it matched polygons of particular feature codes in the source training data set. When a particular template records a single feature code with the highest frequency, it is this class that we assign to a matching polygon that we wish to classify. If two or more feature codes share the greatest frequency of occurrence, we consider that template to be *ambiguous* as we cannot say which class it suggests with greatest probability. We make no feature code assignment in the case of an ambiguous template.

Our content vectors record the number of features of each class neighbouring a polygon. A context structure also describes this, and, in addition, the relationships between those neighbours. This causes CSM to distinguish between different feature neighbourhoods that are identical to the more coarsely grained CVM. In general, we expect CSM templates to be more accurate classifiers, as they are more discerning about what they will match. At the same time, some features requiring classification may not match any CSM template, while a CVM template from the same source training data will match. Thus while CSM cannot match as many features as CVM, it should be a more accurate classifier overall. In order to maximize the number of attempted classifications, we combine the results of CVM and CSM in two ways.

6.3 Combining CVM and CSM Classification Results

In many cases a content vector template will suggest a polygon feature code while CSM cannot because either a matching context structure template isn't found or an ambiguous one is matched. Sometimes, CVM will yield an ambiguous template, while CSM does suggest a class. Other than in these two cases, both techniques will fail or both will suggest a feature code. If the two classifiers disagree, we can either disregard them both and not attempt classification or we can choose one. We amalgamate the feature codes suggested by both the content vector and the context structure classifiers by giving one technique precedence over the other. This yields two classification results in addition to plain CVM and CSM:

CVM>CSM

If the feature code suggested by CVM is chosen when it differs from that propounded by CSM, we say that CVM takes precedence over CSM. We denote this as *CVM>CSM*.

CSM>CVM

If the feature code suggested by CSM is chosen when it differs from that advocated by CVM, we say that CSM takes precedence over CVM. We denote this as *CSM>CVM*.

6.4 Template Quality (training results)

We derived a set of CVM templates and CSM templates from the 36,847 internal topographic polygons in the basNW training data set. Because CSM has $O(n!)$ complexity, in all stages of CSM where graph isomorphism is checked, i.e. identifying context structure templates (training) and matching the training data contexts against those templates (testing), we limited the maximum possible search space to 1 million. This is an arbitrary limit chosen as it was found by trial and error to yield acceptable CSM search times. Beyond this limit no attempt is made to establish a structure match and the corresponding feature is considered to be unmatched. Some statistics on the CVM templates and CSM templates built from the basNW training data are given in Table 6.2.

CVM	CSM	
36,847	36,847	# polygons in training data
36,847	36,847	# polygons represented by templates
5,892	10,518	# templates
~6.25	~3.5	ratio of polygons to templates
4,173	8,438	# templates with sum of frequencies = 1
70.82	80.22	% of templates with sum of frequencies = 1
1,970	1,271	maximum sum of frequencies of any template
1,970	1,271	maximum class frequency of same template

Table 6.2: Template statistics for basNW training data set.

The first row shows the number of features present in the basNW training data. Beneath this is recorded the count of these polygons that was incorporated in a template. CVM exhibits $O(1)$ complexity and can represent any feature as a template. It can be seen that, in this case, the search space limit we set on CSM did not prevent any feature from being incorporated in a template, as all 36,847 polygons were recorded by CSM templates also. In the third row, the number of distinct templates required to cover the data set is shown. Beneath this, we see that there is a ratio of 6.25 features to one CVM template, and a ratio of 3.5 polygons to one CSM template. This is because context structures are a lot more detailed than content vectors and hence require a greater number of templates. In the fifth row, we show the number of templates with a sum of

frequencies of value one i.e. those that matched one feature uniquely. Proportionally, 70.82% of content vectors are unique, while a greater proportion of context structures (80.22%) are once-offs. Again, the difference is explained by CSM going to greater effort to distinguish between neighbourhoods. In the seventh row, we show the maximum sum of frequencies (i.e. the total number of features of any class matched by a specific template) for all templates derived. The last row shows the greatest frequency of occurrence for a single class for those templates. The most frequently occurring content vector had a frequency of 1,970, and all 1,970 of these were multiple surface land. The most frequent context structure, with a frequency of 1,271 matched a feature of that same class in every case. The corresponding templates can be considered “perfect” classifiers because they have estimated probabilities of 100%, based on vast sample sizes. At the other end of the spectrum, templates with a sum of frequencies of one also have an estimated probability of 100%, but we cannot know whether that figure is accurate. The larger the sum of frequencies is for a template, the more confident we can be about the accuracy of its probabilities. A good quality classification template has a high sum of frequencies and a maximum frequency count that is close to that figure.

Figure 6.3 shows a plot of sum of class frequencies against maximum class frequency for the 5,892 content vector templates identified from the training data. These CVM templates have sums of frequencies that are exponentially distributed, 70.82% having a value of one. The straight line represents the ideal template with identical values on both axes, yielding an estimated probability of 100%. It can be seen that as the sum of frequencies increases, templates tend to get closer to this value. Several of the most frequent templates are, in fact, “perfect”. Figure 6.4 presents the same type of plot for the 10,518 context structure templates, and is very similar to the preceding CVM graph. Accordingly, CSM templates have an exponential distribution by sum of frequencies and they tend towards estimated probabilities of 100% as this value increases. A number of the most frequent context structure templates are also “perfect” classifiers. The two outliers near co-ordinates (400,200) correspond to CSM templates that represent almost the precise same set of polygons as those CVM templates that are also outliers near the same point in Figure 6.3.

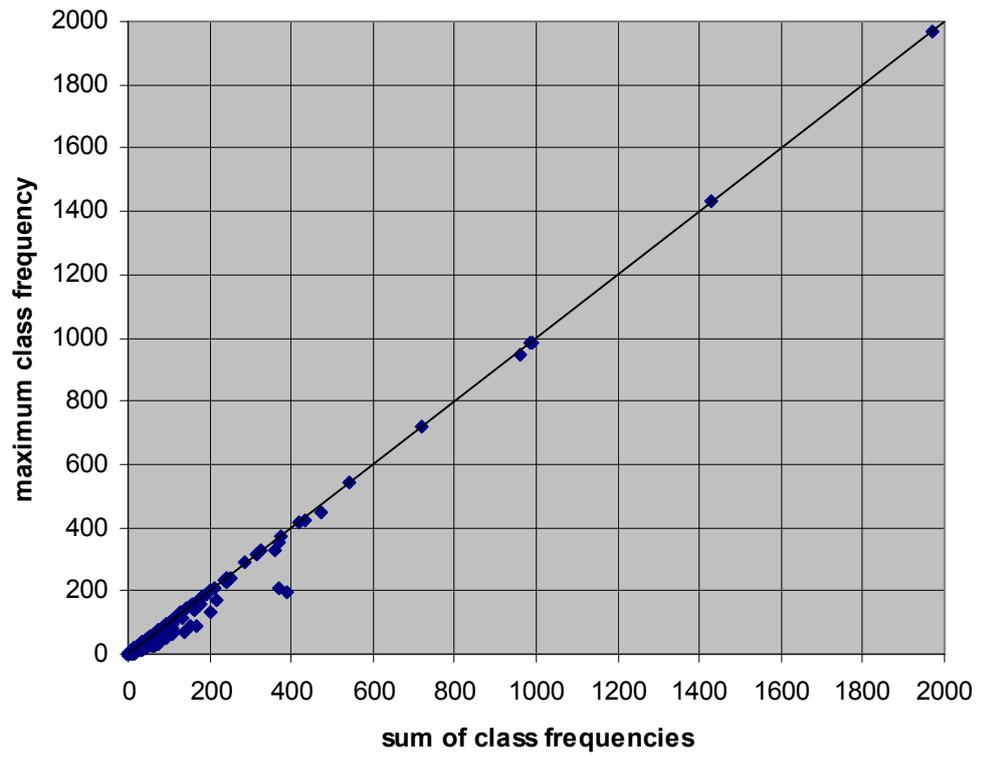


Figure 6.3: Quality of CVM templates from basNW training data.

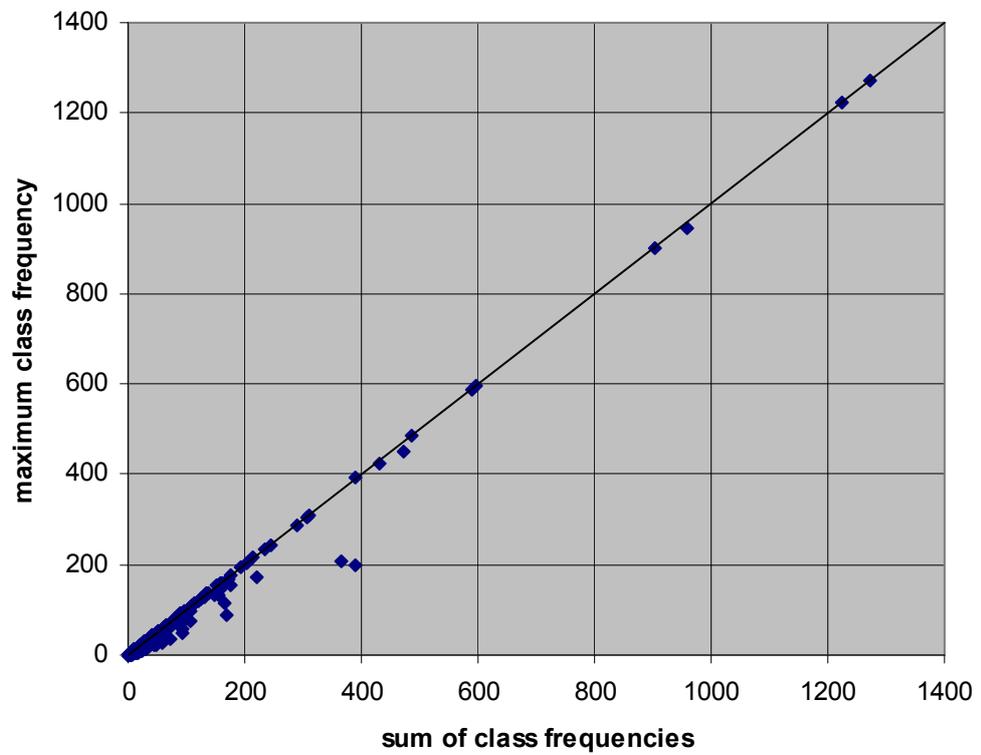


Figure 6.4: Quality of CSM templates from basNW training data.

6.5 Classification Results (testing results)

Having built a set of classification templates from the basNW training data, we proceeded to the testing stage. The contexts of the 5,070 internal topographic polygons in the port testing data set were then identified. Next, we attempted to match each context against the set of templates and thus classify each area feature described using both CVM and CSM if possible.

Table 6.3 presents the results of using the basNW to classify the port data set by using CVM, CSM, CVM>CSM and CSM<CVM. The topmost unshaded rows present the breakdown of the port data set by class, as already presented in Table 6.1. The class upper level of communication is excluded; this is the only feature class present in the training data, but not in the testing data. Beneath these rows the classification results are shown in separate bands for each of the four classification schemes, as labeled down the left. The same categories of figures are given for each scheme, with the overall value for features of all classes being given in bold before being decomposed into the twelve feature classes occurring in the port data set. The first row of figures for each scheme is the number of features unambiguously matched. This is the count of the polygons for which a corresponding template was identified, excluding features whose matching template is ambiguous. For the combined schemes (CVM>CSM, CSM>CVM), the classifier is only considered ambiguous if one of the constituent classifiers (CVM, CSM) is ambiguous while the other is either ambiguous or doesn't match a template. The number of features unambiguously matched is identical for CVM>CSM and CSM>CVM as both require only a single constituent classifier to propose a feature class. The second row of figures for each scheme presents the number of features which were unambiguously matched by a corresponding classifier that propounded the same feature class as the ground truth classifications recorded for each feature in the original OS MasterMap data. This is the count of the features that were correctly classified by the scheme. In the third row of figures the percentage of the total number of features in the testing data that were unambiguously matched by the classifier is noted. In the fourth row of figures, the percentage of the total number of features in the testing data that were correctly classified is given. Lastly, in the fifth row, the percentage of the number of features unambiguously matched by the classifier that were correctly classified by that classifier is presented. This last row of numbers is the most important, because it describes the accuracy of the corresponding classification scheme.

6.5.1 Analysis of Classification Results

Looking again at Table 6.3, the percentage of features unambiguously matched by a classifier is 79.03% overall for CVM and falls to 62.96% for CSM. This is expected because graph-based CSM is more discerning about what it will match than vector-based CVM. The corresponding figures for CVM>CSM and CSM>CVM are identical because both require only one of CVM or CSM to match without ambiguity. The overall number for these combined schemes, 79.45%, is only marginally higher than that of CVM alone. This trend can be seen across all feature classes, and demonstrates that there is a relatively small count of content vector matches that are disambiguated by CSM. For CVM, the number is highest at 96.87% for buildings. This figure only falls to 92.66% for CSM, indicating that the finer granularity of this classifier tends to differentiate between classes other than building. For multiple surface land the number drops from 87.38% for CVM to 43.69%, revealing a high degree of variation in neighbourhood topology for features of this class. The drop-off going from CVM to CSM is not as significant for any other feature class.

class		all classes	building	multiple surface land	general surface	road or track	roadside	natural environment	inland water	general surface step	railway	structure	path	glasshouse
feature code		10021	10053	10056	10172	10183	10111	10089	10054	10167	10185	10123	10062	
total		5070	2138	1284	713	317	296	130	85	39	25	23	17	3
% of data set		100	42.17	25.33	14.06	6.25	5.84	2.56	1.68	0.77	0.49	0.45	0.34	0.06
CVM	# unambiguously matched	4007	2071	1122	396	191	151	19	20	25	4	3	2	3
	# matched & correct	3421	1864	1089	187	178	98	1	2	2	0	0	0	0
	% unambiguously matched	79.03	96.87	87.38	55.54	60.25	51.01	14.62	23.53	64.10	16	13.04	11.76	100
	% total correct	67.48	87.18	84.81	26.23	56.15	33.11	0.77	2.35	5.13	0	0	0	0
	% matched & correct	85.38	90	97.06	47.22	93.19	64.90	5.26	10	8	0	0	0	0
CSM	# unambiguously matched	3192	1981	561	301	163	123	15	16	21	4	3	1	3
	# matched & correct	2744	1786	534	173	157	90	0	2	2	0	0	0	0
	% unambiguously matched	62.96	92.66	43.69	42.22	51.42	41.55	11.54	18.82	53.85	16	13.04	5.88	100
	% total correct	54.12	83.54	41.59	24.26	49.53	30.41	0	2.35	5.13	0	0	0	0
	% matched & correct	85.96	90.16	95.19	57.48	96.32	73.17	0	12.5	9.52	0	0	0	0
CVM>CSM	# unambiguously matched	4028	2081	1124	403	191	152	20	20	25	4	3	2	3
	# matched & correct	3428	1864	1091	191	178	99	1	2	2	0	0	0	0
	% unambiguously matched	79.45	97.33	87.54	56.52	60.25	51.35	15.38	23.53	64.10	16	13.04	11.77	100
	% total correct	67.61	87.18	84.97	26.79	56.15	33.45	0.77	2.35	5.13	0	0	0	0
	% matched & correct	85.1	89.57	97.06	47.39	93.19	65.13	5	10	8	0	0	0	0
CSM>CVM	# unambiguously matched	4028	2081	1124	403	191	152	20	20	25	4	3	2	3
	# matched & correct	3437	1854	1087	210	178	103	1	2	2	0	0	0	0
	% unambiguously matched	79.45	97.33	87.54	56.52	60.25	51.35	15.38	23.53	64.1	16	13.04	11.76	100
	% total correct	67.79	86.72	84.66	29.45	56.15	34.8	0.77	2.35	5.13	0	0	0	0
	% matched & correct	85.33	89.09	96.71	52.11	93.19	67.76	5	10	8	0	0	0	0

Table 6.3: Results achieved by applying classifiers derived from basNW data set to port data set.

Figure 6.5 shows the number of polygons of each feature class correctly classified by each of the four classification schemes. It corresponds to the second row for each scheme in Table 6.3. The figures for CSM are lower than for CVM, in accordance with the number of features unambiguously matched. CSM correctly classifies about half as many multiple surface land features as CVM, due to the latter unambiguously matching about twice as many such land parcels. The improvement of CVM>CSM and CSM>CVM over CVM is only slight, with the latter being ahead overall.

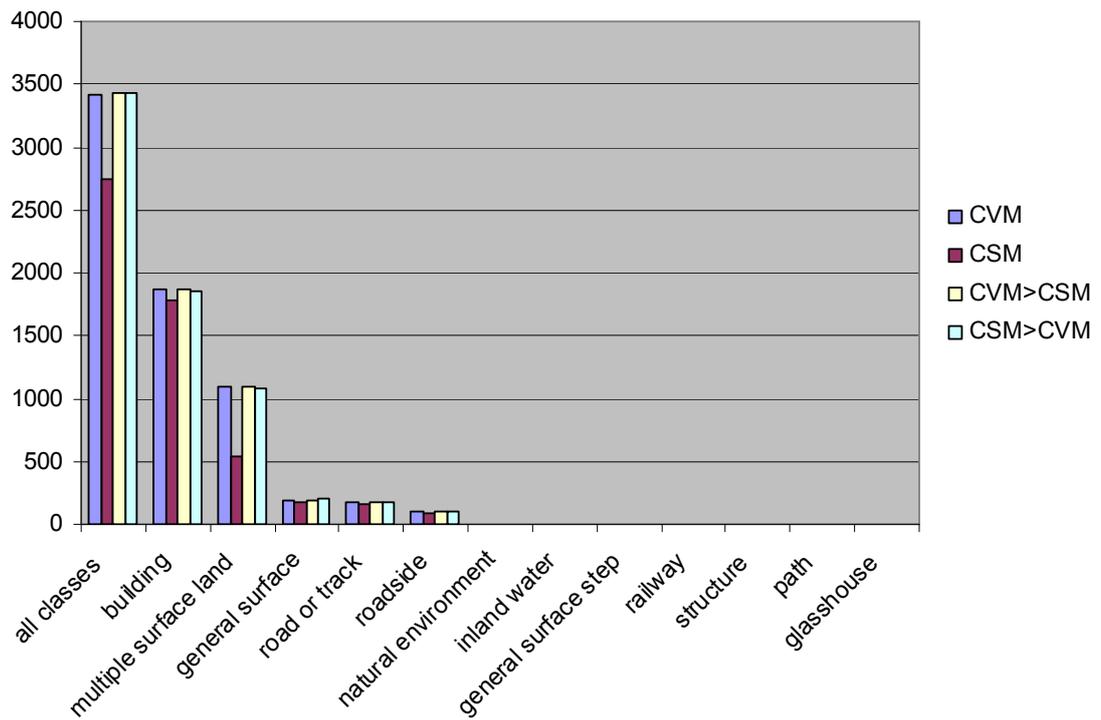


Figure 6.5: Number of polygons of each class correctly classified.

Figure 6.6 presents the percentage of the entire port data set that was correctly classified. It corresponds to row four for each classification scheme in Table 6.3. The features that were not classified correctly include polygons that were misclassified, those which couldn't be classed due to ambiguity, and those for which a template match wasn't established. CVM correctly classified 67.48% of features while CSM achieved 54.12%. The combined classifiers achieved marginally better classification, with CSM>CVM being best overall at 67.79%. This is mostly due to its strong performance with general surface and roadside features. The number of accurate class assignments made by CSM is less than that made by CVM because of the smaller number of features the former matches.

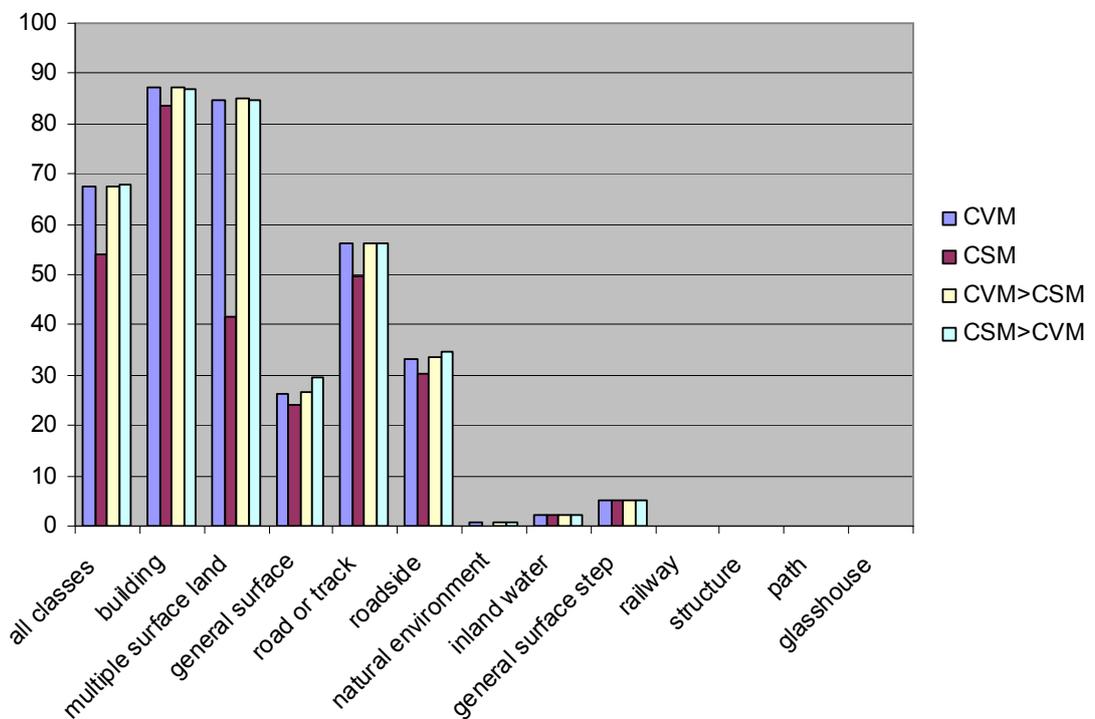


Figure 6.6: Percentage of each class correctly classified.

Figure 6.7 displays the classification accuracy for all of the four schemes. It corresponds to the fifth row for each scheme in Table 6.3. Over all feature classes, CVM achieves an accuracy of 85.38%, while CSM does slightly better at 85.96%. The figure for CVM>CSM and CSM>CVM, at 85.1% and 85.33% respectively, are a slight disimprovement on the accuracy of content vectors alone. The slight improvement in the number of features properly classified by the combined scheme comes at the cost of this reduction in accuracy. CSM>CVM does do a good job of combining the classifiers in the case of general surface and roadside features. Looking at the breakdown by class, accuracy tends to decline from the most common feature types on the left to the rarest on the right. CSM achieves the best result for buildings at 90.16%, for general surface at 57.48%, for road or track at 96.32%, for roadside at 73.17%, for inland water at 12.5% and for general surface step at 9.52%. CVM is most accurate for multiple surface land at 97.06% and natural environment at 5.26%.

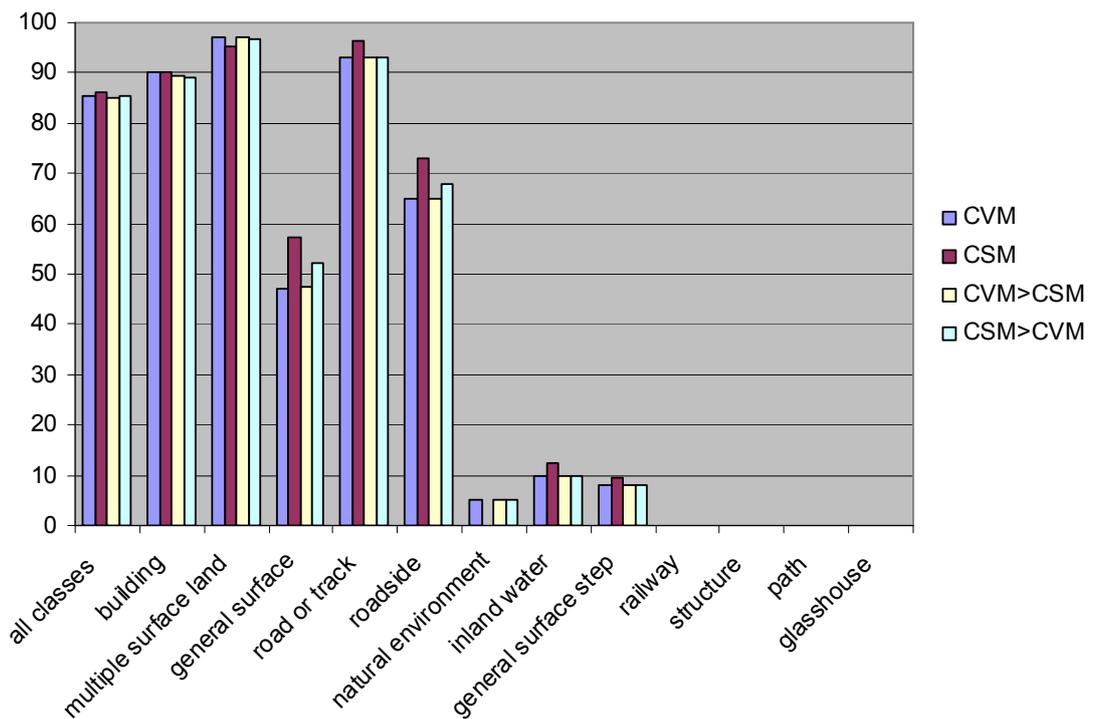


Figure 6.7: Classification Accuracy by class.

6.6 Visualizations of Classifications

In order to give an impression of the feature coding abilities of our classifiers, we present a number of projections of an area of the port testing data set. Figure 6.9 shows this area with all polygons bearing the ground truth classes recorded for this data. Areas within the rectangular frame that are blank either contained features that were filtered out because they were on the perimeter of the data set, or were not included in the corpus to begin with. Figure 6.8 shows the fills used to symbolize the features in this and all subsequent figures.

Figure 6.10 presents the same area symbolized using the classes assigned by the CVM templates generated from the basNW testing data set. Areas that could not be classified, either because a matching template was not identified or because the appropriate template was ambiguous, are not shown. Comparison between it and the preceding figure reveals what features were either not classified, were misclassified, or were correctly classified. Overall, nearly all semi-detached houses (belonging to building class), most multiple surface land and road or track sections were identified. Misclassified large features tend to be disproportionately obvious, such as several areas of general surface land classed as building.

Figure 6.11 shows the same area classified by the CSM templates generated from the basNW training data. As with the preceding CVM output, Semi-detached house classification is very good. Identification of multiple surface land is poorer than for CVM. The number of roads recognized is slightly lower, as is the proportion of misclassified features. Among features that CVM misclassified, but which CSM identified correctly are a number of larger buildings and general surface land.

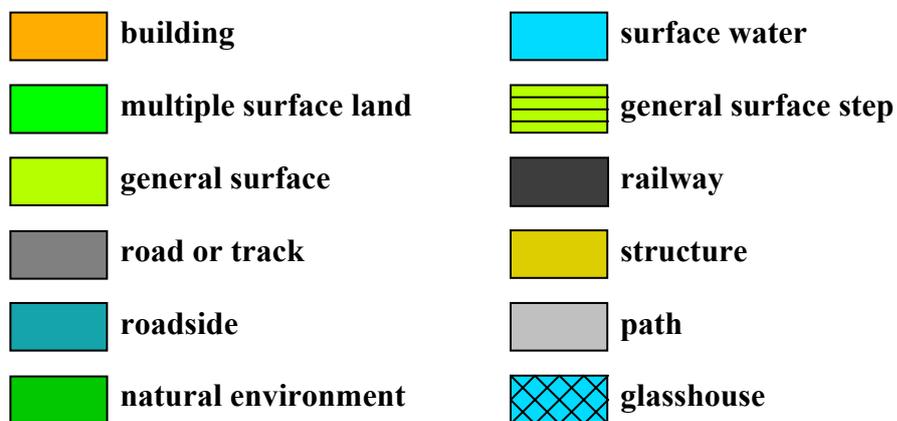


Figure 6.8: Legend for polygon symbolization.

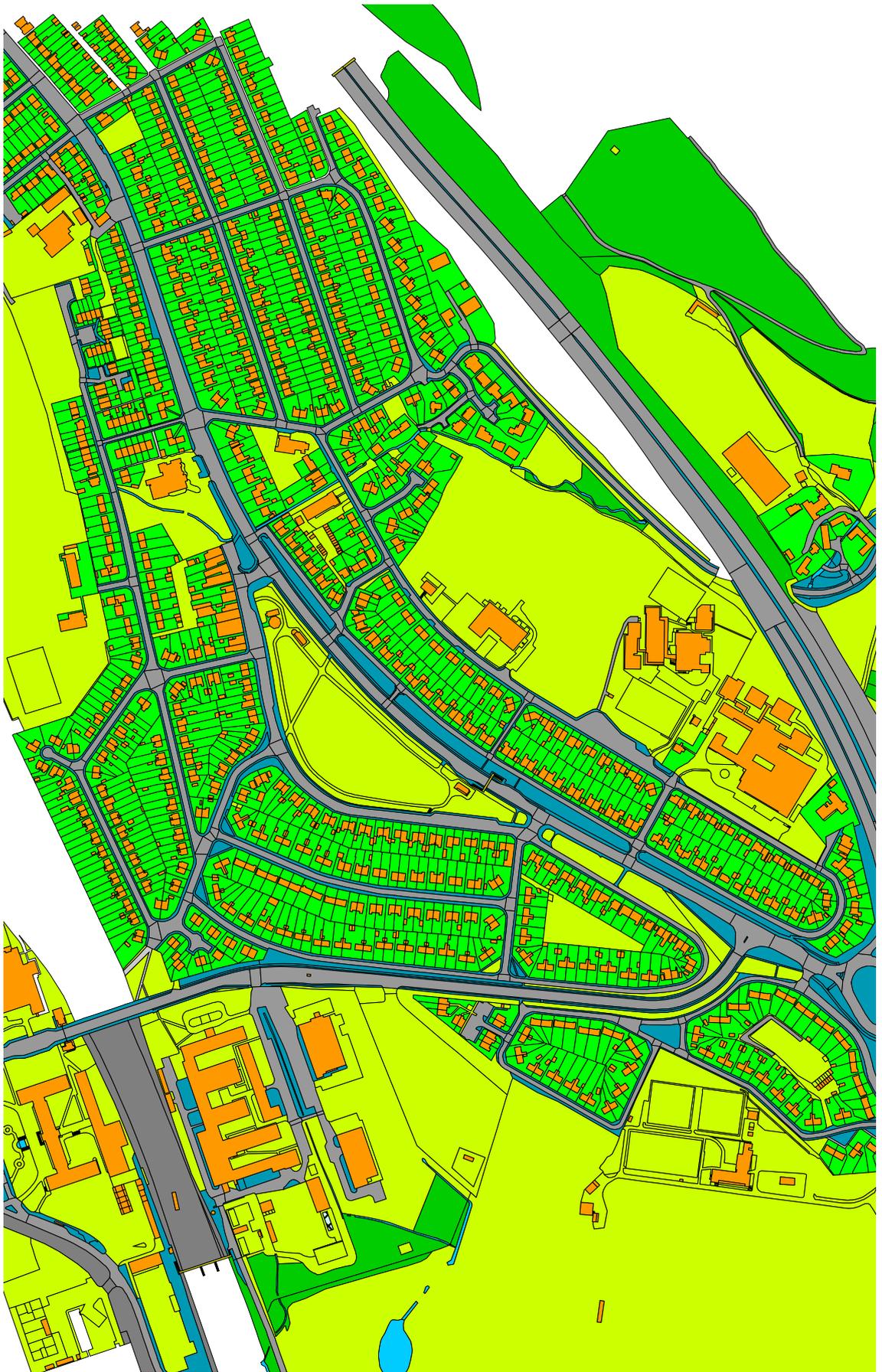


Figure 6.9: Part of port testing data set, showing ground truth classes.



Figure 6.10: Part of port testing data set, showing CVM-assigned classes.



Figure 6.11: Part of port testing data set, showing CSM-assigned classes.

6.6 Conclusion

In this chapter, we described the training and testing of our content vector and context structure polygon classifiers. A large corpus of training data was used to derive a set of these probabilistic classification templates. It was found that 70.82% of CVM templates and 80.22% of CSM templates were unique and consequently unreliable. However, both sets of classifiers had an exponential distribution of sums of frequencies, with the most frequently occurring ones facilitating almost 100% accuracy. The entire set of templates was used to classify a smaller testing data set. Two additional classification results were yielded by combining the output of CVM and CSM. The performance and accuracy of all four classifiers was evaluated by reference to the ground truth feature classes recorded in the testing data. CVM was found to achieve a greater number of accurate classifications than CSM. This is due to the finer granularity of the latter's context matching model. CSM>CVM, which gives precedence to context structures over content vectors, slightly improves the total number of accurate classifications. Due to its greater precision, CSM achieves higher classification accuracy than CVM for the features that it does try to classify. A classification accuracy of 85.96% was achieved across all feature classes, with figures of 90.16% for buildings, 96.32% for road or track and 97.06% for multiple surface land. In the final chapter, we summarize the dissertation, draw conclusions about the significance of the work presented and suggest areas of future work.

7: Conclusion

7.1 Introduction

In this dissertation, we described the successful development and implementation of two context-based polygon classification tools for large-scale topographic vector databases. These were inspired by similarities between map data and geometric analogy problems. Our classifiers establish a match between the neighbourhood of an area feature and those of polygons previously encountered. This analogy facilitates the transfer of classification knowledge from the source domain to the target domain.

In this final chapter, we evaluate the success of our approach, describe some of its limitations and make suggestions to improve on it. We conclude by proposing other areas of cartography and spatial data fields where our technique may also prove useful.

7.2 Evaluation of Context-Based Classification

We developed two tools to classify polygons in topographic data by matching a description of their neighbours against template descriptions of previously encountered area features. A content vector records the number of polygons of each class that bound on an area feature. A context structure records the number of polygons of each class that bound on an area feature, and the relations between those neighbours. To evaluate both techniques, we identified the template content vectors and the template context structures in a large training corpus of topographic polygons. These classification templates were used to assign suggested feature classes to polygons in a testing data set. The results for CVM and CSM were then combined as both CVM>CSM and CSM>CVM. By comparing the class suggestions against the ground truth feature classes recorded in the testing corpus, we were able to evaluate the effectiveness of the individual and the combined classifiers.

7.2.1 Dependence on Pre-Classified Data

Object-based polygon classification techniques such as shape (Keyes & Winstanley, 2001) and texture analysis (Winstanley & Corcoran, 2005) examine individual features in isolation. Their ability to identify a feature's class is independent of any other polygon. In contrast, our context-based techniques rely on the classes of the surrounding area features being known. This is the case for both the training and testing / application

stages. The old adage, “rubbish in, rubbish out” holds true. We cannot identify a feature by its context if we have not encountered features of the same class in identical contexts before. We have specified that high quality data should be used for training context-based classifiers. Such a data set would have an accurate feature class assigned to almost every feature. In addition, a testing corpus needs to be of good quality. This is because a misclassified neighbouring feature would cause us to attempt classification based on a wrong context. Nevertheless, our approach can be used to highlight likely problems, although it might flag the neighbour of a misclassified feature instead. The results of testing any classifier are adversely affected by errors in the ground truth. Errors in data sets are not an issue for our results as we used very high quality data. The dependence on correctly classified neighbours limits our approach to verifying the quality of reasonably well-structured data.

7.2.2 Classification Template Quality

Returning to Table 6.2 in section 6.4, we have shown that limiting the search space for checking graph isomorphism in CSM does not reduce the practical usefulness. In the testing data set of 36,847 polygons, our limit of 1,000,000 trial inter-graph mappings had no impact. The $O(n!)$ complexity of CSM has not restricted its usefulness. Due to the fact that CSM distinguishes between polygon neighbourhoods that CVM considers to be identical, the ratio of polygons to templates in the training data was found to be 6.25 for CVM and 3.5 for CSM. In addition to CSM templates outnumbering CVM templates by about two to one, the former require more space because they need to hold inter-neighbour adjacency information that the latter don't. 70.82% of CVM templates were found to be unique, while the figure for CSM templates was 80.22%. This is another difference attributable to the finer granularity of the latter.

We showed Figures 6.3 and 6.4 that both types of templates exhibit exponential distribution in their total frequencies and that these templates tend towards maximum relative frequencies (i.e. estimated probabilities) of 100% as the total frequency of occurrence increases. This is an important result, as it means that a relatively small number of CVM and CSM templates are extremely accurate classifiers of polygons. In particular, the most frequent CVM template will identify corresponding features as multiple surface land with an estimated probability of $1970/1970=1$. The most matched CSM template identifies the same class of polygons with an estimated probability of $1271/1271=1$. These and several others constitute “perfect” classifiers.

7.2.3 Unmatched Features

The remainder of the figures we reference are taken from Table 6.3 in subsection 6.5.1. Due to the high degree of variation in feature neighbourhoods, many polygons in the testing data did not match against any CVM or CSM template identified during training. Unmatched features are a challenge created by data sparseness and can be addressed by increasing the size of the training data. Because there is no theoretical limit on the number of neighbours a feature may have, it is almost certain that there will be a small proportion of features that would be nationally, and even globally, unique. Other techniques would be required to classify such features. Excluding ambiguous templates, which do not allow classification, CVM matched 79.03% of features in testing, while CSM achieved 62.96%. The higher figure for CVM is due to CSM's inclusion of inter-neighbour adjacencies in its model. This allows CSM to distinguish between features that CVM considers to have identical contexts. CVM matched 96.87% of buildings, while the figure for CSM was 92.66%. These demonstrate that there is a high degree of regularity in the content vectors and the context structures of buildings. The match rates for multiple surface land were 87.38% and 43.69% respectively, revealing that there is far more similarity in the content vectors of these features than in the context structures. This is due to variations in the adjacencies between neighbours.

7.2.4 Classification Accuracy

Each feature that did match a template unambiguously was tentatively assigned the feature class with the greatest relative frequency for that template. By comparing these classes against the ground truth feature code for each polygon, we determined that CVM achieved an accuracy of 85.38%, while CSM scored slightly higher at 85.96%. Because CVM matches a greater number of features than CSM, the number of features correctly classified was 3,421 and 2,744 respectively. Bearing in mind that CVM is an $O(1)$ operation and CSM is $O(n!)$, we can see that the slight improvement in accuracy of CSM over CVM comes at high, but manageable, computational cost and a 20% reduction in output.

The accuracy of CSM was higher across all classes, except natural environment and multiple surface land. This may have been due to inaccuracies in the corresponding CSM template estimated probabilities due to data sparseness. The classification accuracy for building was 90.16% and that for road or track was 96.32%. Where CVM did do better for multiple surface land, it achieved 97.06% accuracy. It is clear that

these three feature classes exhibit a high degree of structural regularity in their neighbourhoods. All three classes have high frequencies in urban areas. They collectively comprise 82.12% of the training and 73.75% of the testing corpora used. Thus, our context-based classifiers are well suited to the majority of urban data.

7.2.5 Combined Classifier Performance

We combined the output of our two classification schemes in two separate ways. CVM>CSM gives precedence to CVM when there is a disagreement. Reciprocally, CSM>CVM gives preference to CSM in a conflict. Both techniques achieved an unambiguous match rate of 79.45%, a slight improvement over CVM's 79.03%. The overall accuracy across all feature classes was 85.38% for CVM, 85.96% for CSM, 85.1% for CVM>CSM and 85.33% for CSM>CVM. It had been expected that the combined classifiers would benefit from the increased number of correct classifiers achieved by CVM and the greater accuracy of CSM, and that the accuracy figures would fall between both. Instead, the accuracy of CVM>CSM and CSM>CVM was lower than both CVM and CSM. This indicates that when one classifier was ambiguous the other tended to either suggest a classification based on a template with a low total feature code frequency (inaccurate due to data sparseness) or a feature class that was only marginally dominant.

7.2.6 Summary & Recommendations

Both CVM and CSM successfully classify the most common types of features found in urban data. Mapping of built-up areas is the most commonly used and valuable of topographic data. CSM does not classify as many features as CVM, but it is slightly more accurate for most feature classes. The templates used by both techniques require a large frequency of occurrence in order to yield accurate feature class probabilities. Both CVM and CSM templates exhibit logarithmic distribution in their total frequencies. The templates with the highest frequencies tend to suggest one of the more common feature classes with near or actual 100% estimated probability. That is, high frequency templates tend to be excellent classifiers. A limited set of these building, multiple surface land and road or track templates could be selected and used in isolation to achieve feature code assignment or verification of an extremely high quality. Alternatively, classification tools can be used to identify features that may have improper feature codes and flag these to a user, suggesting a new class assignment. The

user could either accept the suggestion, or take a different action. Any approach that seeks to combine the output of CVM & CSM needs to take into account the total frequency of the templates, and the highest class frequency. This is because proportional frequencies can be misleading for the less common templates. Increasing training data size should eliminate many cases of error resulting from data sparseness.

7.3 Importance of Land Use

Context-based classification has proven most accurate at identifying the building, multiple surface land and road or track features that predominate in urban data. Different built-up areas exhibit variations in polygon topology. For instance, town and city centres tend to have terraced housing and shops. There may be path or general surface routes between and behind some of the buildings. Certain terraced houses may face directly on to roadside, not having a front garden. Most of these buildings and their associated multiple surface land areas will have topologies and hence contexts that differ from those of suburbia, where gardens tend to enclose semi-detached or detached homes. Rural areas tend to be without path features, with a far lower density of houses, many of which are part of general surface farmyards. Fields vary enormously in their contexts, while many natural environment polygons are even more complex in their size and relationships to other features. Context-based classifiers trained on suburban data cannot be expected to perform well in other areas, especially in the countryside.

A wide array of land types need to be used for training CVM and CSM templates to allow them to maximize their usefulness. While context-based classification was inspired by the topological regularity seen in suburban data, there seems to be far fewer structural patterns in rural data. We suspect that object-based techniques would be better suited to classifying such area features. It is possible to use feature density to classify broad regions as urban, suburban or rural. If separate sets of classifiers were generated from and applied to the different regions, improvements might be achieved in classification.

7.4 Future Work

There are a number of ways that the work presented here could be extended to improve the quality of topographic polygon data.

7.4.1 Combining Output with Other Classifiers

Our context-based classifiers are one of several feature coding techniques implemented within our research group. Others include shape-based analysis, statistical language model classifiers and aerial photography texture-based classification. Different feature coding approaches will achieve higher accuracy for specific feature classes. By combining the classification results from the varying techniques, it is expected that an overall accuracy greater than any single approach alone can be achieved. The feature code probabilities generated by our classifiers approximate its confidence in each possible feature class assignment. This could temper the weight given to its input in a combined, multi-discipline classification approach.

7.4.2 Improving Accuracy

At present, we use the feature code frequencies recorded by our classification templates to estimate the probability that a feature matching that template is of a particular class. This approach is vulnerable to inaccuracy caused by data sparseness. For instance, if a particular template was found to have only a single match in a training corpus, the proportional frequency, and hence the estimated probability of the associated feature code is $1/1=1.0$. If that template was matched a hundred times in a wider training data set, we might find that most instances do indeed match that same class, making the estimate appropriate. However, we might find that the original feature is the only one of its class to match and that all ninety-nine other analogues exhibit an identical, but differing class. If an estimated feature code probability of $100/100=1.0$ was recorded from the smaller training corpus, it is far less likely that this value would decrease when we widen our training data. Accordingly, the higher template frequency counts are, the more confident we can be about our probability estimates.

The inaccuracy resulting from data sparseness could be addressed by setting a minimum template total frequency, below which the template is not trusted. This raises the question of where we set cut-off point. Alternatively, we could attenuate all probability estimates based on the total frequency of each template. For instance,

$100/100=1.0$ might be rescaled to 0.99, and $1/1=1.0$ might be lowered to 0.5. Finally, we could specify a minimum estimated probability required to allow classification. Any of these approaches, or a combination of them, would reduce the number of features correctly classified, whilst increasing accuracy.

7.4.3 Searching For Known Classification Artifacts

In addition to using our classifiers to verify the feature code accuracy of as many polygons as possible, they could be applied to identify known categories of misclassification in topographic data. For instance, a building completely surrounded by another building is a very unusual occurrence. Such a feature is likely to be a courtyard, which would be classed as general surface. Other usual occurrences would be a building adjacent to water, a building adjacent to a road, or a road unconnected to other roads. Such errors might arise from incorrect feature codes in line data or errors in the automated structuring process employed to build a polygon layer from that data.

7.4.4 Extending Context Model beyond Immediate

Neighbours

Our model of a polygon's context is restricted to those area features bordering on the feature described. This could be extended to incorporate the neighbours of these neighbours and so on. A content vector could have an additional pair of vectors added to record the feature classes of each subsequent ring of outlying polygons. Further graph isomorphism search complexity restrictions would need to be placed on CSM to allow it to extend.

7.4.5 Incorporation of Other Topographic Layers in Context Model

Our context model is restricted to analyzing the topology of area features surrounding a particular polygon. We could include cartographic polygons, line and point features that intersect the topographic polygon being described. These can yield insight into the likely class of associated area features. For instance, slopes, cliffs and pylons (all cartographic features in OS MasterMap) are unlikely to occur outside of natural environment, general surface, roadside or railway areas. Individual trees are usually only recorded in areas of natural environment or general surface. It is unlikely that any topographic (real world) feature will intersect a surface water or tidal water area. Such rules could be useful in narrowing down classification possibilities and in identifying feature coding errors. The topographic line layer is the source from which the OS MasterMap polygon layer is derived. For instance, an area contained within a building outline is classified as a building, roads are bounded by road or track edges. Investigation of line features might identify classification errors introduced during the generation of the polygon layer.

7.5 Applications of Work

In addition to improving the classification accuracy of area features in large-scale topographic vector data, our neighbourhood context approach could be applied to identifying features in other types of spatial data.

7.5.1 Topographic Vector Line Data

As with all feature layers in topographic vector data, line features are attributed their own feature codes. These line features intersect at the nodes where they meet and occasionally cross each other or coincide (e.g. an administrative boundary following a stream). These relationships are analogous to the adjacency relations between topographic polygons. Line features will also exhibit a degree of structure e.g. a building edge must join with other building edges or building boundaries unless it encloses a region by itself, a road edge must join with other road edges. Modeling the context of line features may allow the identification of misclassified lines.

7.5.2 Small-Scale Topographic Data

Our context-based classifiers have been applied to the most precise and detailed of all topographic data, large-scale vector data. Increasingly, smaller scale data sets are derived from such large-scale holdings through a variety of cartographic generalization algorithms. These operate iteratively to reduce the clutter that results when one zooms out from a map. Among the techniques used are the merging of nearby features of the same class together, the simplification of shapes and the removal of certain features when they become too small or have insufficient priority to be shown at a particular scale. The aim is a legible map covering large areas in a smaller scale which shows the most important features. Given a large-scale polygon layer, the feature codes of the areas can be used to classify regions in the resulting smaller-scale data. A context-based classification process could be applied to small-scale data to verify the accuracy of the generalization process. Small-scale vector data that is gathered via survey rather than derived from more detailed mapping may also be quality checked.

7.5.3 Raster Data Classification

Raster data can also be considered to be composed of classified objects. Every pixel in a raster image has a colour value that corresponds to the class of features that location belongs to. A continuous area of pixels with the same colour may be considered to be an object. There are two types of adjacency between pixels in raster data: 4-adjacency and 8-adjacency. Four-adjacency exists between a pixel and those cells bounding it above, below, and to either side. This is analogous to line-adjacency in polygon vector data. Eight-adjacency is exhibited by the aforementioned four bordering pixels and the four other cells that surround the central pixel, but only touch at the corners. This latter group of four pixels that are 8-adjacent to, but not 4-adjacent to the central pixel are the equivalent of point-adjacent neighbours in polygon vector data. Continuous areas of a single colour may thus be considered to be area features that either are or aren't adjacent to other features. Context-based classification can be used to classify these areas, potentially improving the quality of a vast array of raster spatial data types.

7.5.4 Classification of Features in Architectural Plans

In addition to geographic data, spatially referenced vector data is used in *computer-aided design (CAD)* programs to design buildings, aircraft, cars and consumer goods, amongst other things. Architectural drawings follow a logic: walls sit on foundations, windows sit on sills, panes lie in windows, doors sit in frames, and light fixtures connect to wiring, which in turn connects to switches and other wires. Each of these features is classified and symbolized in architectural plans. By observing the topology of these features, the rules that underpin a valid infrastructural plan (if not a structural plan!) can be modeled as feature contexts and these can be used to verify other plans, highlighting errors. This would become increasingly important for larger buildings, such as sky-scrapers, where a vast number of floor plans are required.

7.5.5 Verifying Circuit Diagrams

Printed circuit boards (PCBs) are used in nearly all electronic goods. They usually resemble green wafers and are covered in copper electrical pathways that connect the electrical components that they support. They range from simple components in a torch, to intricately detailed multi-layered *motherboards* in a contemporary computer. The costs of designing, verifying and building the necessary *masks* to produce a new PCB can be huge. PCBs are designed in CAD software, from which the necessary masks are built. By modeling the topology of the circuits and the components they connect to, it might be possible to identify errors in vastly complex PCB designs, saving the time and cost of building unsuccessful prototypes. This would have a positive impact on time to market and the cost of many electrical and electronic goods.

7.6 Conclusion

Content vector matching and context structure matching have proven successful in verifying the feature classes of polygons in large-scale topographic databases. CVM can classify a greater variety of features, while CSM is capable of greater accuracy. Both are well suited to identifying the houses, gardens and roads common in suburban areas. These features possess a high degree of structural regularity and generate several high quality classification templates. Greater accuracy is possible by limiting the use of low frequency templates. Context-based classification's focus on neighbouring features makes it well suited to verifying the integrity of high quality topographic data. Training across a large corpus of data will minimize data sparseness related misclassification, while training on a variety of urban and rural areas will allow a greater variety of features to be recognized. There are several avenues to improving classification accuracy that may be explored. Context-based classification has the potential to make valuable contributions to data quality in several other spatial data fields.

Bibliography

- Bohan, A. & O'Donoghue, D.** (2000) "A Model for Geometric Analogies using Attribute Matching", AICS-2000 - 11th Artificial Intelligence and Cognitive Science Conference, Aug. 23-25, NUI Galway, Ireland.
- Evans, T. G.** (1968) A Program for the Solution of a Class of Geometric-Analogy Intelligence Test Questions. *In: M. Minsky ed. Semantic Information Processing.* MIT Press.
- Falkenhainer, B., Forbus, K. D. & Gentner, D.** (1989) Structure-mapping engine. *Artificial Intelligence.* 41, 1-63.
- Gentner, D.** (1983) *Structure-Mapping: A Theoretical Framework for Analogy,* *Cognitive Science,* 7, 155-170.
- Harley, J.B. & Woodward, D.** eds. (1987) *The History of Cartography: Cartography in Prehistoric, Ancient and Medieval Europe and the Mediterranean,* vol. 1. Chicago, University of Chicago Press, pp. 113-115.
- Hofstadter, D.** (1995) *Fluid Concepts and Creative Analogies.* Basic Book, NY.
- Intelligent Systems Laboratory.** (2001) SICStus Prolog User's Manual, release 3.8.7. Swedish Institute of Computer Science. Kista, Sweden.
- Jones, C.B.** (1997) *Geographical Information Systems and Computer Cartography.* Harlow, England, Longman.
- Keane, M. T. Ledgeway, T., & Duff, S.** (1994) Constraints on Analogical Mapping: A Comparison of Three Models. *Cognitive Science,* 18, 387-438.
- Keyes, L. & Winstanley, A. C.** (2001) "Using Moment Invariants for Classifying Shapes on Large Scale Maps", *Journal of Computers, Environment and Urban Systems* 25(1), 119-130, January 2001.
- Longley, P. A., Goodchild, M. F., Maguire, D. J. & Rhind, D. W.** (2005) *Geographical Information Systems and Science.* 2nd ed. Wiley.
- Malory, Thomas.** (2000) *Le Morte d'Arthur.* Ed. Matthews, J. Illustrated by Ferguson, Anna-Marie. London, Cassell.
- Marinilli, M., Micarelli, A., & Sciarrone, F.** (1999) *A Case-Based Approach to Adaptive Information Filtering for the WWW,* in Proc. of the 2nd Workshop on Adaptive Systems and User Modeling on the World Wide Web, Sixth International Conference on User Modeling UM-99, Banff, Canada, June 20-24 1999.

- O'Donoghue, D** (2004) *Finding Novel Analogies*. Thesis (Ph.D.). University College Dublin
- Ordnance Survey** (2005) OS MasterMap User guide v6. Ordnance Survey, Southampton, England. 03/2005
- OED.** (1989) "analogy" The Oxford English Dictionary. (2nd ed. 1989). OED Online. [online] Oxford University Press. Available from: <http://dictionary.oed.com/cgi/entry/50007888> [Accessed 31 October 2007].
- OED.** (2001) "metaphor, n." OED Online. December 2001 [online]. Oxford University Press. Available from: <http://dictionary.oed.com/cgi/entry/00307429> [Accessed 31 October 2007].
- Peuquet, D.** (1984) A conceptual framework and comparison of data models. *Cartographica*, **21**(4), pp. 66-113.
- Salaik, B.** (2004) Statistical Language Models for Topographic Data Recognition. Thesis (M.Sc.). National University of Ireland, Maynooth.
- Sidney, H.** (1994) Camelot, The Theodore H. White Interview with Jackie Kennedy. In: Kennedy Assassination Chronicles. Fall 2005 Vol. 1 issue 3. JFK Lancer Productions & Publications. Available from: <http://jfklander.com/pdf/Camelot.pdf> [Accessed 31 October 2007]
- Sobel, D.** (1996) Longitude. London, Fourth Estate.
- Stellhorn, T.,** (2000) Tap Built-in Functionality for Better Performance. *ArcUser* [online], Jul-Sep 2000, 30-31. Available from: <http://www.esri.com/news/arcuser> [Accessed 31 October 2007]
- Tomai, E., Lovett, A., Forbus, K., & Usher, J.** (2005). A Structure Mapping Model for Solving Geometric Analogy Problems. *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, Stresa, Italy, 2190-2195.
- Waner, S. & Costenoble, S.R.** (2001) Introduction to Logic. [online] Last updated September 2001. Available from: http://people.hofstra.edu/stefan_waner/Realworld/logic/logicintro.html [Accessed 31 October 2007]
- White, T.H.** (1963) For President Kennedy: An Epilogue. *Life* 12/6/63
- Wikipedia** (2007a) *King Arthur* http://en.wikipedia.org/wiki/King_arthur [Accessed 31 October 2007]
- Wikipedia** (2007b) *Le Morte D'Arthur* http://en.wikipedia.org/wiki/Le_Morte_Darthur [Accessed 31 October 2007]
- Wikipedia** (2007c) *Theodore White* http://en.wikipedia.org/wiki/Theodore_White [Accessed 31 October 2007]

Winstanley, A. C. & Corcoran, P. (2005) 'Large Robust Texture Separability within Aerial Photography' In Proceedings of GIS Research UK, edited by Billen, R., Drummond, J., Forest, D. and Jaoa E., University of Glasgow, pp 93-98.